

Discount
Discount ID
Discount Status
Discount Type
Discount Value
Discount Start Date
Discount End Date

Organization
Organization ID
Organization Name
Pipeline Indicator
Pipeline Manufacturer ID
Pipeline Subscription Code
Description
Address Line 1
Address Line 2
City
State or Province
Zip or Postal Code
Country

Organization Product Line Discount

Organization Commodity Discount

Organization Part Discount

Product Line
Product Line ID
Product Line Number
Product Line Name
Brief Description
Description
Image URL
Sequence 1
Sequence 2
Sequence 3
Active Indicator
Long Description

Commodity
Commodity ID
Commodity Number
Commodity Name
Brief Description
Description
Suggested List Price
Image URL
Sequence 1
Sequence 2
Sequence 3
Active Indicator
Long Description

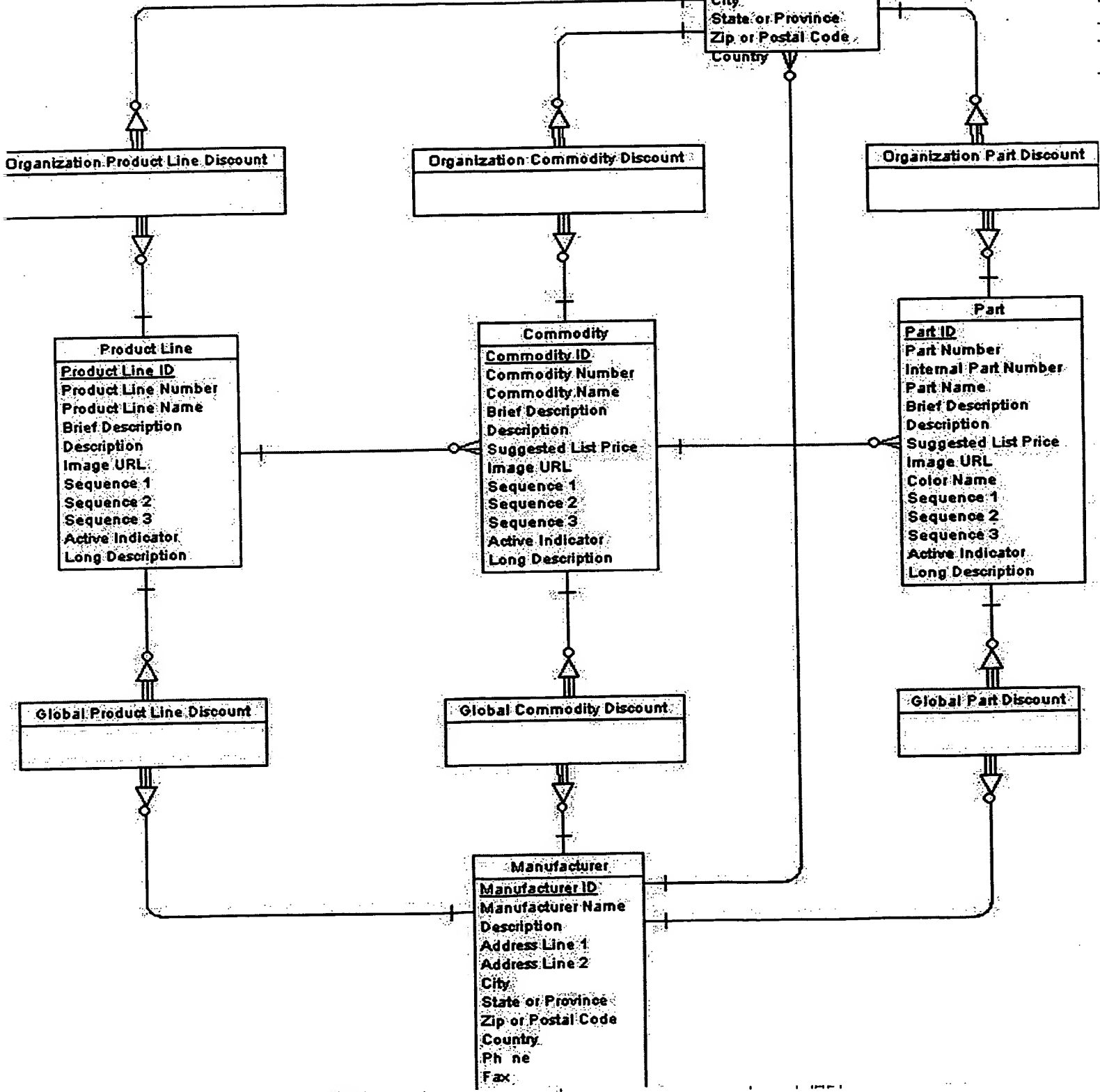
Part
Part ID
Part Number
Internal Part Number
Part Name
Brief Description
Description
Suggested List Price
Image URL
Color Name
Sequence 1
Sequence 2
Sequence 3
Active Indicator
Long Description

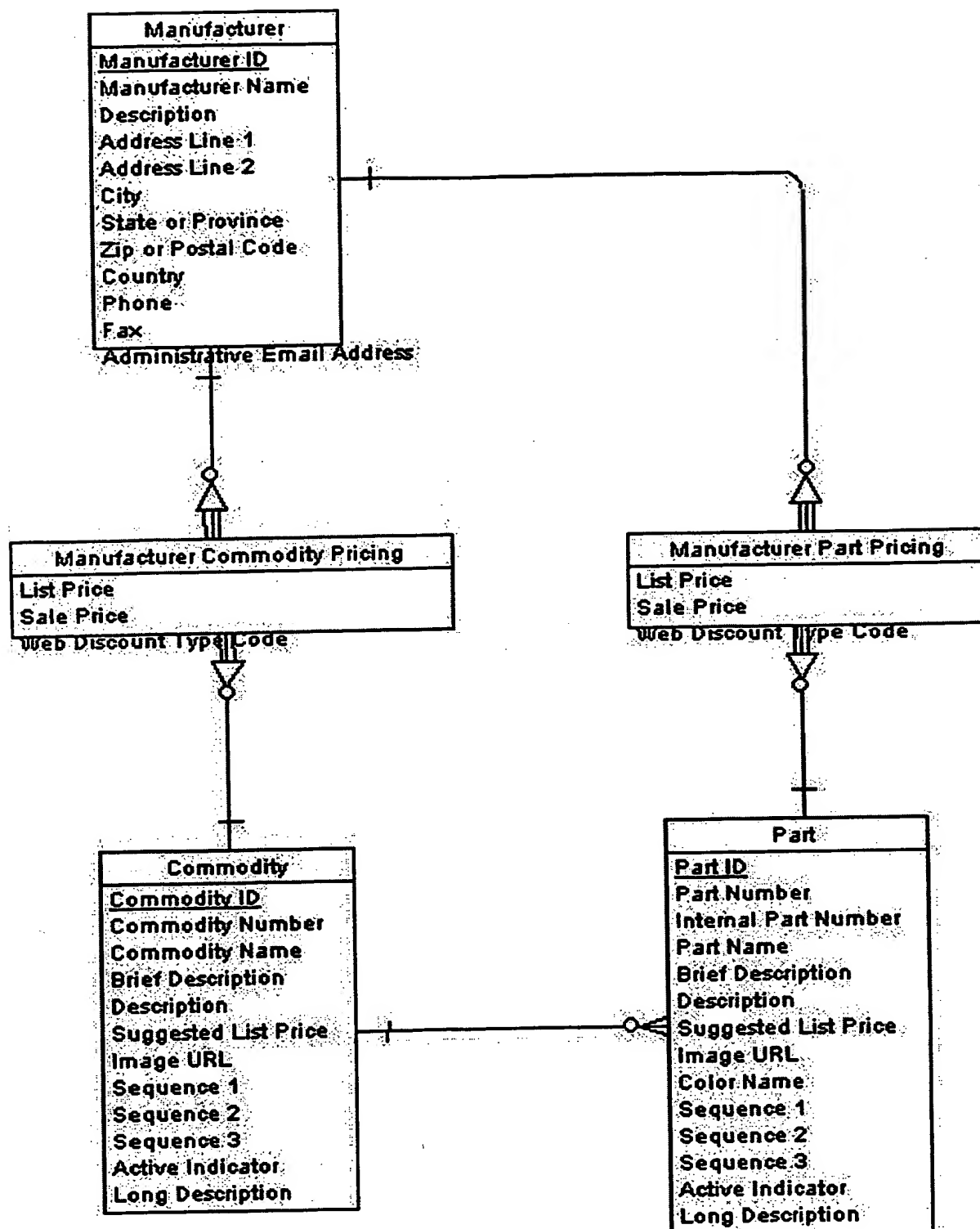
Global Product Line Discount

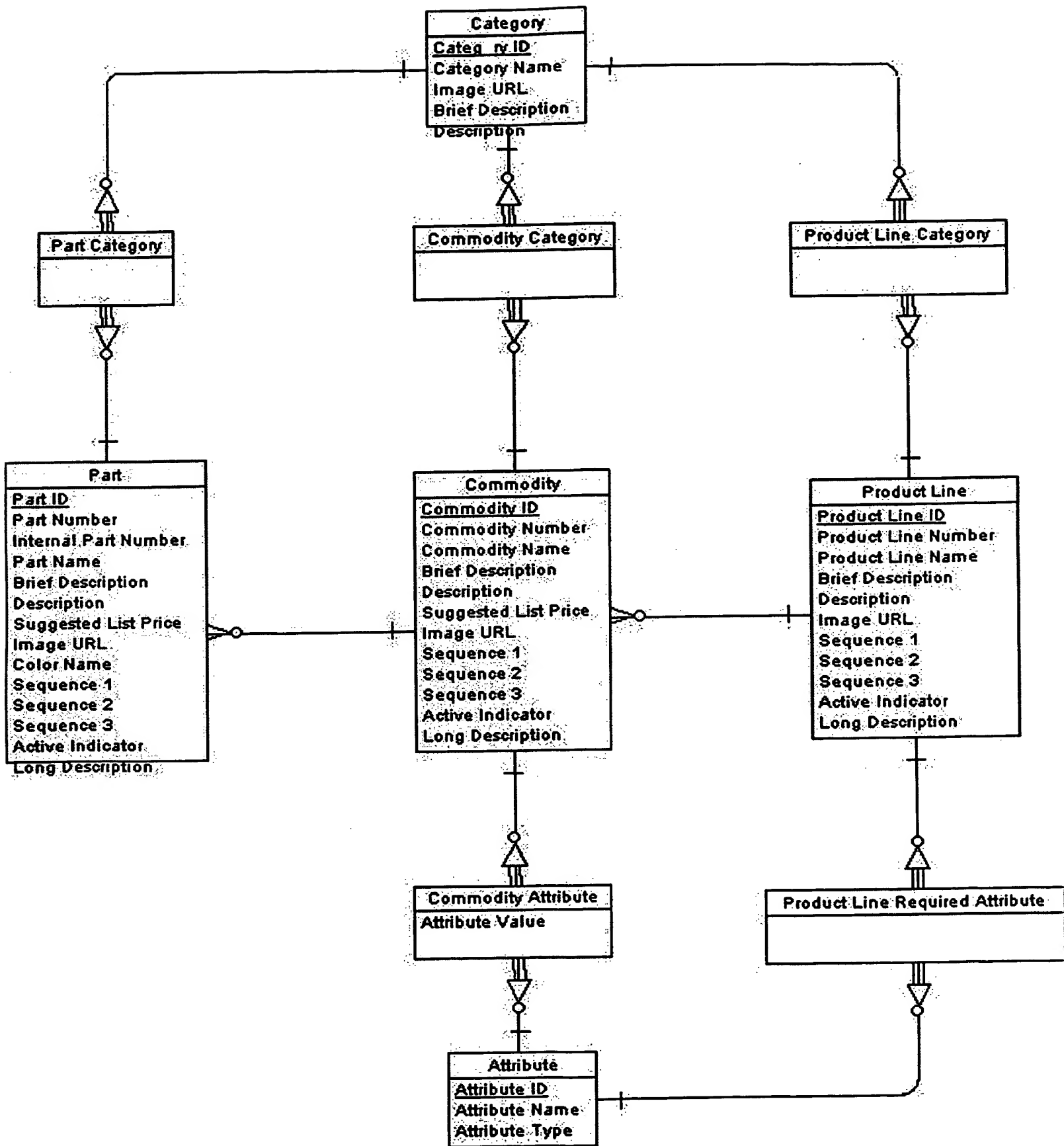
Global Commodity Discount

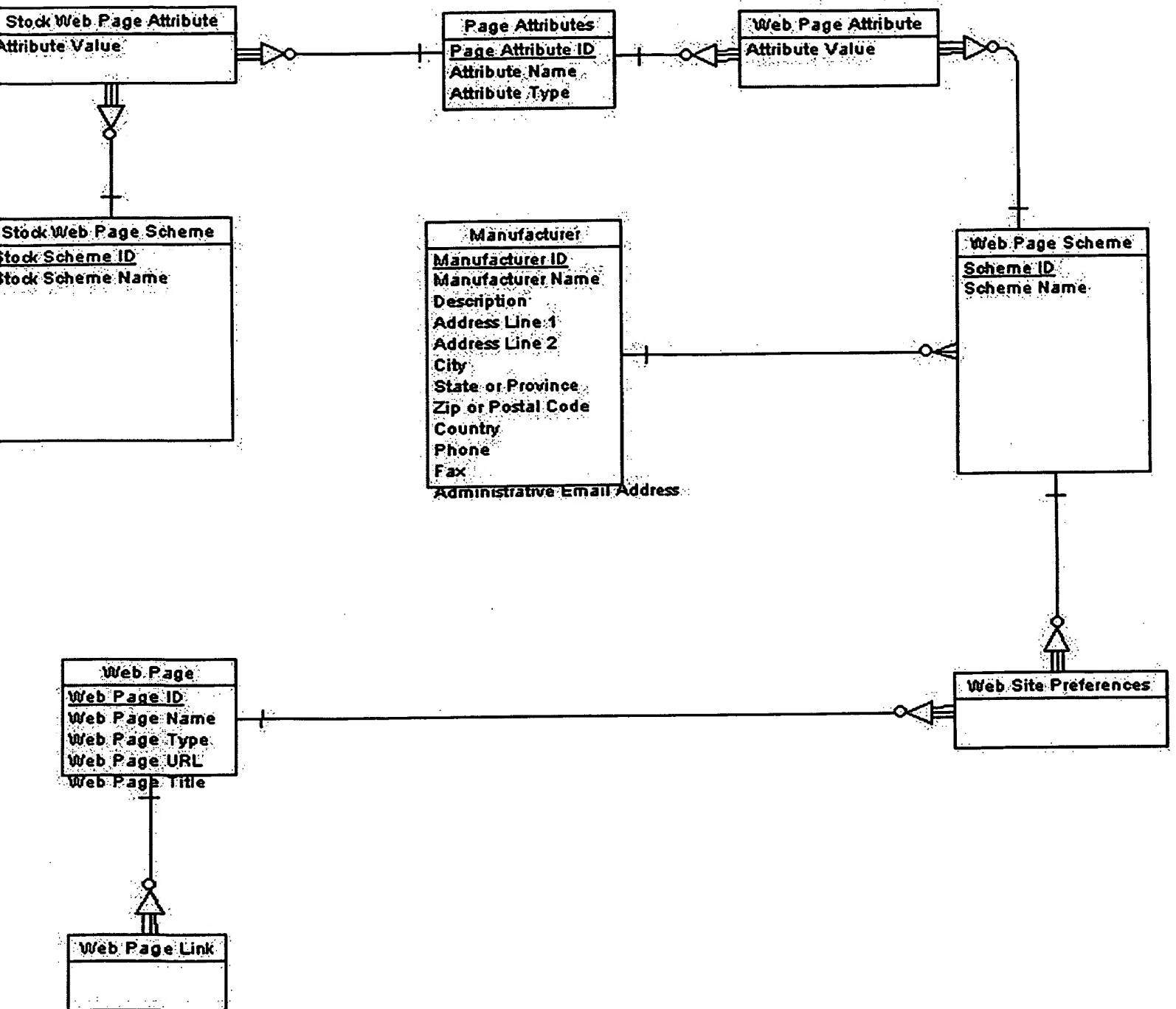
Global Part Discount

Manufacturer
Manufacturer ID
Manufacturer Name
Description
Address Line 1
Address Line 2
City
State or Province
Zip or Postal Code
Country
Phone
Fax









Accessories

Column List

Cod	Type
commodity_id	int
descr	varchar(255)
create_dtm	datetime

Address

Column List

Code	Type
addr_id	int
addr_typ_cd	char(1)
entity_id	int
entity_typ_cd	varchar(5)
company_nme	varchar(255)
nme	varchar(255)
addr_1	varchar(50)
addr_2	varchar(50)
city	varchar(30)
st_prov	varchar(30)
zip_pstl_cd	varchar(10)
country	varchar(25)
phone	varchar(15)
ext	varchar(10)
alt_phone	varchar(15)
alt_ext	varchar(10)
active_ind	char(1)
create_dtm	datetime
last_maint_dtm	datetime

Arced Text

Column List

Code	Type
design_id	int
obj_id	int
obj_nme	varchar(50)
x_pos	float
y_pos	float
width	float
height	float
font_id	int
style_id	int
font_size	int
text_algn	char(1)
text_str	varchar(255)
pos_lck_ind	char(1)

Code	Typ
size_lck_ind	char(1)
cont_lck_ind	char(1)
style_lck_ind	char(1)

Attribute

Column List

Code	Type
attr_id	int
attr_nme	varchar(50)
attr_typ	char(1)
create_dtm	datetime
last_maint_dtm	datetime

Attribute Code

Column List

Code	Type
code_id	int
attr_id	int
create_dtm	datetime

Available Billing Type

Column List

Code	Type
mfr_id	int
bill_typ_id	int
create_dtm	datetime

Available Shipping Type

Column List

Code	Type
mfr_id	int
ship_typ_id	int
create_dtm	datetime

Base URL

Column List

Code	Type
base_url_id	int
mfr_id	int

Code	Type
base_url	varchar(255)
create_dtm	datetime
last_maint_dtm	datetime

Billing Type

Column List

Code	Type
bill_typ_id	int
bill_brief_descr	varchar(255)
descr	varchar(255)
bill_typ_cd	char(1)
create_dtm	datetime
last_maint_dtm	datetime

Border

Column List

Code	Type
design_id	int
obj_id	int
border_obj_id	int
obj_nme	varchar(50)
shape_typ_cd	char(1)
line_weight	int
marg_width_pts	int
style_lck_ind	char(1)

Category

Column List

Code	Type
cat_id	int
cat_nme	varchar(50)
img_url	varchar(255)
brief_descr	varchar(255)
descr	varchar(255)
long_descr	varchar(6800)
active_ind	char(1)
create_dtm	datetime
last_maint_dtm	datetime

Code

Column List

Code	Typ
------	-----

Code	Type
code_id	int
code_name	varchar(255)
rgb_val	int
create_dtm	datetime
last_maint_dtm	datetime

Commodity

Column List

Code	Type
commodity_id	int
prod_line_id	int
commodity_num	varchar(50)
commodity_nme	varchar(50)
brief_descr	varchar(255)
descr	varchar(255)
sugg_list_price	money
img_url	varchar(255)
stamp_face_id	int
seq_1	int
seq_2	int
seq_3	int
active_ind	char(1)
long_descr	varchar(6800)
create_dtm	datetime
last_maint_dtm	datetime

Commodity Attribute

Column List

Code	Type
commodity_id	int
attr_id	int
attr_val	varchar(255)
create_dtm	datetime
last_maint_dtm	datetime

Commodity Category

Column List

Code	Type
cat_id	int
commodity_id	int
create_dtm	datetime

Country

Column List

Code	Type
country_id	int
country_nme	varchar(25)
create_dtm	datetime
last_maint_dtm	datetime

Default Available Part**Column List**

Code	Type
mfr_id	int
part_id	int
commodity_id	int
prod_line_id	int
create_dtm	datetime

Design**Column List**

Code	Type
design_id	int
mfr_id	int
org_id	int
usr_id	int
stamp_face_id	int
templ_cat_id	int
rev_num	int
design_typ_cd	char(1)
border_design_id	int
border_obj_id	int
templ_nme	varchar(255)
design_cmnt	varchar(7700)
create_dtm	datetime
last_maint_dtm	datetime

Font**Column List**

Code	Type
font_id	int
font_face_nme	varchar(50)
create_dtm	datetime
last_maint_dtm	datetime

Font Style

Column List

C d	Typ
font_id	int
style_id	int
create_dtm	datetime

Global Commodity Discount**Column List**

Code	Type
mfr_id	int
commodity_id	int
disc_id	int
disc_stus_cd	char(1)
disc_typ_cd	char(1)
disc_val	money
disc_start_dte	datetime
disc_end_dte	datetime
create_dtm	datetime
last_maint_dtm	datetime

Global Part Discount**Column List**

Code	Type
mfr_id	int
part_id	int
disc_id	int
disc_stus_cd	char(1)
disc_typ_cd	char(1)
disc_val	money
disc_start_dte	datetime
disc_end_dte	datetime
create_dtm	datetime
last_maint_dtm	datetime

Global Product Line Discount**Column List**

Code	Type
mfr_id	int
prod_line_id	int
disc_id	int
disc_stus_cd	char(1)
disc_typ_cd	char(1)
disc_val	money
disc_start_dte	datetime
disc_end_dte	datetime
create_dtm	datetime

Code	Type
last_maint_dtm	datetime

Horizontal Line

Column List

Code	Type
design_id	int
obj_id	int
line_width_pctg	int
line_just	char(1)
size_lck_ind	char(1)

Image

Column List

Code	Type
design_id	int
obj_id	int
obj_nme	varchar(50)
x_pos	float
y_pos	float
width	float
height	float
img_scale	int
img_src	varchar(255)
pos_lck_ind	char(1)
size_lck_ind	char(1)
cont_lck_ind	char(1)

Job Title

Column List

Code	Type
job_title_id	int
job_title_descr	varchar(255)
create_dtm	datetime
last_maint_dtm	datetime

Layout Box

Column List

Code	Type
design_id	int
obj_id	int
bor_design_id	int
bor_obj_id	int

Code	Type
obj_nme	varchar(50)
layout_typ_cd	char(1)
shape_typ_cd	char(1)
x_pos	float
y_pos	float
width	float
height	float
anchor_pt	int
width_pctg	int
pos_lck_ind	char(1)

Layout Element

Column List

Code	Type
design_id	int
obj_id	int
elem_seq_num	int
elem_typ_cd	int
elem_obj_id	int
space_before_pts	int
space_after_pts	int

Line

Column List

Code	Type
design_id	int
obj_id	int
obj_nme	varchar(50)
start_x_pos	float
start_y_pos	float
end_x_pos	float
end_y_pos	float
line_weight	int
pos_lck_ind	char(1)
style_lck_ind	char(1)

Manufacturer

Column List

Code	Type
mfr_id	int
mfr_nme	varchar(50)
descr	varchar(255)
addr_1	varchar(50)
addr_2	varchar(50)
city	varchar(30)

C de	Typ
st_prov	varchar(30)
zip_pstl_cd	varchar(10)
country	varchar(25)
phone	varchar(15)
fax	varchar(15)
admin_email_addr	varchar(255)
ord_email_addr	varchar(255)
ord_refresh_rate	int
ss_ind	char(1)
pl_mfr_ind	char(1)
unattend_oper_ind	char(1)
use_alt_img_ind	char(1)
home_site_url	varchar(255)
active_ind	char(1)
create_dtm	datetime
last_maint_dtm	datetime

Manufacturer Accepted Part

Column List

Code	Type
mfr_id	int
part_id	int
src_mfr_id	int
prod_line_id	int
commodity_id	int
accept_ind	char(1)
dflt_src_ind	char(1)
avail_ind	char(1)
create_dtm	datetime
last_maint_dtm	datetime

Manufacturer Accepted Products Reference

Column List

Code	Type
mfr_id	int
part_id	int
src_mfr_id	int
prod_line_id	int
commodity_id	int
mfr_ts_cd	char(1)
prod_line_ts_cd	char(1)
comm_ts_cd	char(1)
accept_ind	char(1)
dflt_src_ind	char(1)
avail_ind	char(1)
src_price	money

Manufacturer Commodity Pricing

Column List

Code	Type
mfr_id	int
commodity_id	int
list_price	money
sale_price	money
web_disc_typ_cd	char(1)
web_disc_val	money
create_dtm	datetime
last_maint_dtm	datetime

Manufacturer Default Scheme

Column List

Code	Type
mfr_id	int
schem_id	int
create_dtm	datetime
last_maint_dtm	datetime

Manufacturer Employee

Column List

Code	Type
usr_id	int
mfr_id	int

Manufacturer Last Used Order Number

Column List

Code	Type
mfr_id	int
ord_num	int
create_dtm	datetime
last_maint_dtm	datetime

Manufacturer Part Pricing

Column List

Code	Type
mfr_id	int
part_id	int
list_price	money
sale_price	money

Code	Type
web_disc_typ_cd	char(1)
web_disc_val	money
create_dtm	datetime
last_maint_dtm	datetime

Manufacturer Product Line Pricing

Column List

Code	Type
mfr_id	int
prod_line_id	int
web_disc_typ_cd	char(1)
web_disc_val	money
create_dtm	datetime
last_maint_dtm	datetime

Manufacturer Products Reference

Column List

Code	Type
mfr_id	int
part_id	int
prod_line_id	int
commodity_id	int
prod_line_ts_cd	char(1)
comm_ts_cd	char(1)

Manufacturer Shipping Rate

Column List

Code	Type
mfr_id	int
ship_typ_id	int
min_amt	money
ship_cost	money
create_dtm	datetime
last_maint_dtm	datetime

Manufacturer Tax Rate

Column List

Code	Type
st_prov_id	int
mfr_id	int
tax_pctg	money
create_dtm	datetime

Code	Type
last_maint_dtm	datetime

Manufacturer Web Page Content

Column List

Code	Type
web_pg_cont_typ_id	int
mfr_id	int
cont_txt	varchar(8000)
create_dtm	datetime
last_maint_dtm	datetime

Manufacturer Web Page Layout

Column List

Code	Type
mfr_id	int
layout_id	int
create_dtm	datetime

Member

Column List

Code	Type
usr_id	int
mfr_id	int
org_id	int

Member Products Reference

Column List

Code	Type
mfr_id	int
part_id	int
prod_line_id	int
commodity_id	int
prod_line_ts_cd	char(1)
comm_ts_cd	char(1)
list_price	money
sale_price	money
special_price	money
web_disc	money
final_price	money
disc_ind	char(1)
price_reduct_label	varchar(100)
price_reduct_val	money

Code	Type
price_reduct_typ_cd	char(1)
web_disc_label	varchar(100)
web_disc_val	money
web_disc_typ_cd	char(1)

Order Header

Column List

Code	Type
ord_id	int
usr_id	int
ord_src_mfr_id	int
org_id	int
mfr_id	int
ord_num	int
ord_dte	datetime
ord_stus_cd	char(1)
ord_stop_stus_cd	char(1)
on_behalf_of_mfr_id	int
drop_ship_ind	char(1)
ship_from_mfr_id	int
po_num	varchar(30)
ref_num	varchar(30)
pl_queue_id	int
sold_to_frst_nme	varchar(30)
sold_to_lst_nme	varchar(30)
sold_to_company_nme	varchar(255)
sold_to_addr_1	varchar(50)
sold_to_addr_2	varchar(50)
sold_to_city	varchar(30)
sold_to_st_prov	varchar(30)
sold_to_zip_pstl_cd	varchar(10)
sold_to_country	varchar(25)
ship_to_nme	varchar(255)
ship_to_company_nme	varchar(255)
ship_to_addr_1	varchar(50)
ship_to_addr_2	varchar(50)
ship_to_city	varchar(30)
ship_to_st_prov	varchar(30)
ship_to_zip_pstl_cd	varchar(10)
ship_to_country	varchar(25)
ship_to_phone	varchar(15)
ship_to_ext	varchar(10)
bill_to_nme	varchar(30)
bill_to_company_nme	varchar(255)
bill_to_addr_1	varchar(50)
bill_to_addr_2	varchar(50)
bill_to_city	varchar(30)
bill_to_st_prov	varchar(30)
bill_to_zip_pstl_cd	varchar(10)
bill_to_country	varchar(25)
bill_to_phone	varchar(15)
bill_to_ext	varchar(10)

C de	Typ
bill_to_alt_phone	varchar(15)
bill_to_alt_ext	varchar(10)
cc_nme	varchar(50)
cc_num	varchar(16)
cc_exp_dte	char(4)
bill_typ_cd	char(1)
bill_brief_descr	varchar(255)
bill_descr	varchar(255)
ship_brief_descr	varchar(255)
ship_descr	varchar(255)
calc_tax	money
calc_ship_cost	money
shop_url	varchar(255)
create_dtm	datetime
last_maint_dtm	datetime

Order History

Column List

Code	Type
ord_hist_id	int
ord_id	int
usr_id	int
ord_evt_dtm	datetime
ord_stus_cd	char(1)
ord_email_sent_ind	char(1)
descr	varchar(500)
email_addr	varchar(255)
email_trk_cd	int
ip_addr	varchar(29)

Order Item History

Column List

Code	Type
ord_item_hist_id	int
ord_id	int
item_num	int
item_evt_dtm	datetime
item_stus_cd	char(2)
descr	varchar(255)
email_addr	varchar(255)
ip_addr	varchar(29)

Order Line Item

Column List

Code	Type
------	------

Code	Type
ord_id	int
item_num	int
part_id	int
pl_ord_id	int
pl_item_num	int
item_stus_cd	char(2)
item_stop_stus_cd	char(2)
src_mfr_id	int
ord_qty	int
ship_qty	int
ship_dte	datetime
ship_track_num	varchar(30)
ret_qty	int
unit_price	money
design_id	int
create_dtm	datetime
last_maint_dtm	datetime

Order Notes

Column List

Code	Type
ord_notes_id	int
ord_hist_id	int
note_typ_cd	char(2)
notes	varchar(500)
create_dtm	datetime

Order Notes Archive

Column List

Code	Type
ord_id	int
note_typ_cd	char(1)
notes	varchar(8000)
create_dtm	datetime
last_maint_dtm	datetime

Organization

Column List

Code	Type
org_id	int
mfr_id	int
org_nme	varchar(50)
pl_ind	char(1)
pl_mfr_id	int
pl_subscript_cd	varchar(20)

Code	Type
descr	varchar(255)
addr_1	varchar(50)
addr_2	varchar(50)
city	varchar(30)
st_prov	varchar(30)
zip_pstl_cd	varchar(10)
country	varchar(25)
phone	varchar(15)
fax	varchar(15)
email_addr	varchar(255)
use_web_disc_ind	char(1)
use_dflt_ship_cost_ind	char(1)
use_alt_img_ind	char(1)
active_ind	char(1)
create_dtm	datetime
last_maint_dtm	datetime

Organization Available Billing Type

Column List

Code	Type
org_id	int
bill_typ_id	int
create_dtm	datetime

Organization Available Part

Column List

Code	Type
org_id	int
part_id	int
commodity_id	int
prod_line_id	int
create_dtm	datetime

Organization Available Shipping Type

Column List

Code	Type
org_id	int
ship_typ_id	int
create_dtm	datetime

Organization Commodity Discount

Column List

Cod	Type
org_id	int
commodity_id	int
disc_id	int
disc_stus_cd	char(1)
disc_typ_cd	char(1)
disc_val	money
disc_start_dte	datetime
disc_end_dte	datetime
create_dtm	datetime
last_maint_dtm	datetime

Organization Part Discount

Column List

Code	Type
part_id	int
org_id	int
disc_id	int
disc_stus_cd	char(1)
disc_typ_cd	char(1)
disc_val	money
disc_start_dte	datetime
disc_end_dte	datetime
create_dtm	datetime
last_maint_dtm	datetime

Organization Product Line Discount

Column List

Code	Type
org_id	int
prod_line_id	int
disc_id	int
disc_stus_cd	char(1)
disc_typ_cd	char(1)
disc_val	money
disc_start_dte	datetime
disc_end_dte	datetime
create_dtm	datetime
last_maint_dtm	datetime

Organization Products Reference

Column List

Code	Type
org_id	int
part_id	int
prod_line_id	int

C de	Typ
commodity_id	int
prod_line_ts_cd	char(1)
comm_ts_cd	char(1)
list_price	money
sale_price	money
special_price	money
web_disc	money
final_price	money
disc_ind	char(1)
price_reduct_label	varchar(100)
price_reduct_val	money
price_reduct_typ_cd	char(1)
web_disc_label	varchar(100)
web_disc_val	money
web_disc_typ_cd	char(1)

Organization Shipping Rate

Column List

Code	Type
org_id	int
ship_typ_id	int
min_amt	money
ship_cost	money
create_dtm	datetime
last_maint_dtm	datetime

Oval

Column List

Code	Type
design_id	int
obj_id	int
obj_nme	varchar(50)
x_pos	float
y_pos	float
width	float
height	float
line_weight	int
pos_lck_ind	char(1)
size_lck_ind	char(1)
style_lck_ind	char(1)

Page Attributes

Column List

Code	Type
page_attr_id	int

Code	Type
attr_nme	varchar(50)
attr_typ	char(1)
create_dtm	datetime
last_maint_dtm	datetime

Part

Column List

Code	Type
part_id	int
commodity_id	int
part_num	varchar(50)
int_part_num	varchar(50)
part_nme	varchar(50)
brief_descr	varchar(255)
descr	varchar(255)
sugg_list_price	money
img_url	varchar(255)
color_nme	varchar(255)
seq_1	int
seq_2	int
seq_3	int
active_ind	char(1)
long_descr	varchar(6800)
create_dtm	datetime
last_maint_dtm	datetime

Part Category

Column List

Code	Type
cat_id	int
part_id	int
create_dtm	datetime

Pipeline Queue

Column List

Code	Type
pl_queue_id	int
usr_id	int
mfr_id	int
src_mfr_id	int
po_num	varchar(30)
ship_to_addr_id	int
ship_to_nme	varchar(255)
ship_to_company_nme	varchar(255)
ship_to_addr_1	varchar(50)

Cod	Typ
ship_to_addr_2	varchar(50)
ship_to_city	varchar(30)
ship_to_st_prov	varchar(30)
ship_to_zip_pstl_cd	varchar(10)
ship_to_country	varchar(25)
ship_to_phone	varchar(15)
ship_to_ext	varchar(10)
temp_ship_addr_sel_ind	char(1)
bill_to_addr_id	int
cc_nme	varchar(50)
cc_num	varchar(16)
cc_exp_dte	char(4)
bill_typ_id	int
bill_typ_cd	char(1)
bill_brief_descr	varchar(255)
bill_descr	varchar(255)
incl_all_note_ind	char(1)
incl_drop_ship_note_ind	char(1)
create_dtm	datetime
last_maint_dtm	datetime

Pipeline Queue Item

Column List

Code	Type
pl_queue_id	int
ord_id	int
item_num	int
drop_ship_ind	char(1)
ship_typ_id	int
ship_brief_descr	varchar(255)
ship_descr	varchar(255)
pl_notes	varchar(500)
int_notes	varchar(500)
email_text	varchar(500)

Pipeline Queue Notes

Column List

Code	Type
pl_queue_id	int
pl_note_typ_cd	char(1)
notes	varchar(500)

Product Line

Column List

Code	Type
------	------

Code	Type
prod_line_id	int
prod_line_num	varchar(50)
prod_line_nme	varchar(50)
brief_descr	varchar(255)
descr	varchar(255)
img_url	varchar(255)
seq_1	int
seq_2	int
seq_3	int
active_ind	char(1)
long_descr	varchar(6800)
create_dtm	datetime
last_maint_dtm	datetime

Product Line Category

Column List

Code	Type
cat_id	int
prod_line_id	int
create_dtm	datetime

Product Line Required Attribute

Column List

Code	Type
prod_line_id	int
attr_id	int
create_dtm	datetime

Rectangle

Column List

Code	Type
design_id	int
obj_id	int
obj_nme	varchar(50)
x_pos	float
y_pos	float
width	float
height	float
line_weight	int
pos_lck_ind	char(1)
size_lck_ind	char(1)
style_lck_ind	char(1)

Sequence Log

Column List

Code	Typ
seq_num	int
table_nme	varchar(50)
last_used_id	int
create_dtm	datetime
last_maint_dtm	datetime

S ssion**Column List**

Code	Type
session_id	varchar(25)
session_info	varchar(8000)
create_dtm	datetime

Shipping Type**Column List**

Code	Type
ship_typ_id	int
ship_brief_descr	varchar(255)
descr	varchar(255)
create_dtm	datetime
last_maint_dtm	datetime

Shopping Basket Item**Column List**

Code	Type
usr_id	int
item_seq_num	int
part_id	int
qty	int
unit_price	money
design_id	int
create_dtm	datetime
last_maint_dtm	datetime

Stamp Face**Column List**

Code	Type
stamp_face_id	int
stamp_face_nme	varchar(50)
shape_typ_cd	char(1)
advt_width_in	float

Code	Type
advt_width_mm	float
advt_height_in	float
advt_height_mm	float
act_width_mm	float
act_height_mm	float
marg_width_mm	float
excl_area_x_mm	float
excl_area_y_mm	float
excl_width_mm	float
excl_height_mm	float
excl_area_loc_cd	char(1)
create_dtm	datetime
last_maint_dtm	datetime

State And Province

Column List

Code	Type
st_prov_id	int
country_id	int
st_prov_nme	varchar(30)
st_prov_abbrev	varchar(2)
create_dtm	datetime
last_maint_dtm	datetime

Stock Web Page Attribute

Column List

Code	Type
stk_schm_id	int
page_attr_id	int
attr_val	varchar(255)
create_dtm	datetime
last_maint_dtm	datetime

Stock Web Page Scheme

Column List

Code	Type
stk_schm_id	int
stk_schm_nme	varchar(50)
create_dtm	datetime
last_maint_dtm	datetime

Supplier Administrator

Column List

Cod	Type
usr_id	int

Template Category**Column List**

Code	Type
templ_cat_id	int
templ_cat_nme	varchar(255)
create_dtm	datetime
last_maint_dtm	datetime

Text**Column List**

Code	Type
design_id	int
obj_id	int
obj_nme	varchar(50)
x_pos	float
y_pos	float
width	float
height	float
font_id	int
style_id	int
font_size	int
text_algn	char(1)
text_str	varchar(255)
pos_lck_ind	char(1)
size_lck_ind	char(1)
cont_lck_ind	char(1)
style_lck_ind	char(1)

User**Column List**

Code	Type
usr_id	int
title	varchar(10)
frst_nme	varchar(30)
mid_init	varchar(1)
lst_nme	varchar(30)
company_nme	varchar(255)
email_addr	varchar(255)
alt_email_addr	varchar(255)
job_title	varchar(50)
pswd	varchar(12)
pswd_hint	varchar(50)

Code	Type
chlng_phrss	varchar(255)
chlng_resp	varchar(255)
user_typ_cd	char(1)
addr_1	varchar(50)
addr_2	varchar(50)
city	varchar(30)
st_prov	varchar(30)
zip_pstl_cd	varchar(10)
country	varchar(25)
phone	varchar(15)
ext	varchar(10)
alt_phone	varchar(15)
alt_ext	varchar(10)
fax	varchar(15)
active_ind	char(1)
create_dtm	datetime
last_maint_dtm	datetime

Web Page

Column List

Code	Type
web_pg_id	int
web_pg_nme	varchar(50)
web_pg_typ_cd	char(1)
web_pg_url	varchar(255)
web_pg_title	varchar(255)
create_dtm	datetime
last_maint_dtm	datetime

Web Page Attribute

Column List

Code	Type
schm_id	int
page_attr_id	int
attr_val	varchar(255)
create_dtm	datetime
last_maint_dtm	datetime

Web Page Content Type

Column List

Code	Type
web_pg_cont_typ_id	int
web_pg_cont_typ	varchar(50)
create_dtm	datetime
last_maint_dtm	datetime

Web Page Layout

Column List

Code	Type
layout_id	int
layout_typ_cd	char(1)
layout_nme	varchar(50)
layout_src	varchar(7964)
create_dtm	datetime
last_maint_dtm	datetime

Web Page Link

Column List

Code	Type
web_pg_id	int
usr_id	int
create_dtm	datetime

W b Page Scheme

Column List

Code	Type
schr_id	int
mfr_id	int
schr_nme	varchar(30)
create_dtm	datetime
last_maint_dtm	datetime

Web Site Preferences

Column List

Code	Type
web_pg_id	int
schr_id	int
create_dtm	datetime

BOrder.cls

VERSION 1.0 CLASS

BEGIN

```
MultiUse = -1 'True
Persistable = 0 'NotPersistable
DataBindingBehavior = 0 'vbNone
DataSourceBehavior = 0 'vbNone
MTSTransactionMode = 2 'RequiresTransaction
```

END

```
Attribute VB_Name = "BOrder"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = True
Option Explicit
Option Base 0
```

```
Private Enum emEmail
    emAddOrder = 1
    emStopOrder = 2
    emUpdateOrderStatus = 3
    emUpdateOrderItemStatus = 4
End Enum
```

Private m_colEmail As Collection

'Note

'=====

'All the functions that start with Int do the actual processing for the function without the Int. The moving of the code was done to have a wrapper around the function for e-mail porposes.

'=====

'=====

'this function adds an order

'input - all the fields in the ORD_HDR table

' record set with the new order

'=====

```
Public Function AddOrder( _
    ByVal alUserID As Long, ByVal alOrgID As Long, ByVal alManfID As Long, _
    ByVal alShipFromManfID As Long, ByVal asPONumber As String, _
    ByVal asRefNumber As String, ByVal asShipToName As String, _
    ByVal asShipToAddr1 As String, ByVal asShipToAddr2 As String, _
    ByVal asShipToCity As String, ByVal asShipToState As String, ByVal asShipToZip
As String, _
    ByVal asShipToCountry As String, ByVal asBillToName As String, ByVal
asBillToAddr1 As String, _
    ByVal asBillToAddr2 As String, ByVal asBillToCity As String, ByVal asBillToState
As String, _
    ByVal asBillToZip As String, ByVal asBillToCountry As String, ByVal
asCreditcardNumber As String, _
    ByVal asCreditCardExpiryDate As String, ByVal asDropShipIndicator As String, _
    ByVal asBillTypeCD As String, ByVal asShipBriefDescr As String, ByVal acTax As
Currency, _
    ByVal acShippingCost As Currency, ByVal asInternalNotes As String, _
    ByVal asEmail As String, ByVal asIPAddress As String, ByVal asPipelineNotes As
String, _
    ByVal asShippingInstructions As String, ByVal asShipToPhone As String, ByVal
asShipToExt As String, _
    ByVal asBillToPhone As String, ByVal asBillToExt As String, ByVal
asBillToAltPhone As String, _
    ByVal asBillToAltExt As String, ByVal asBillBriefDescr As String, _
```

```

Border.cls
ByVal asBillDescr As String, ByVal asShipDescr As String, _
ByVal asShipToCompanyNme As String, ByVal asBillToCompanyNme As String,
ByVal alOnBehalfOfManfID As Long, ByVal asCreditcardName As String, ByVal asURL
As String) As ADODB.Recordset

```

```

On Error GoTo ErrHandler

'var
Dim oRS As ADODB.Recordset

Set oRS = IntAddOrder( _
    aluserID, -1, alOrgID, alManfID, alShipFromManfID, asPONumber, asRefNumber,
    _
    asShipToName, asShipToAddr1, asShipToAddr2, asShipToCity, asShipToState,
asShipToZip, asShipToCountry, _
    asBillToName, asBillToAddr1, asBillToAddr2, asBillToCity, asBillToState,
asBillToZip, asBillToCountry, _
    asCreditcardNumber, asCreditCardExpiryDate, asDropShipIndicator, _
    asBillTypeCD, asShipBriefDescr, acTax, acShippingCost, asInternalNotes, _
    asEmail, asIPAddress, asPipelineNotes, asShippingInstructions,
asShipToPhone, _
    asShipToExt, asBillToPhone, asBillToExt, asBillToAltPhone, asBillToAltExt,
asBillBriefDescr, _
    asBillDescr, asShipDescr, asShipToCompanyNme, asBillToCompanyNme, 0,
asCreditcardName, asURL)

'exit
Set AddOrder = oRS

SendEmails

CtxSetComplete

Set oRS = Nothing
Exit Function

```

```

ErrHandler:
CtxSetAbort
Set oRS = Nothing
RaiseError TypeName(Me), "AddOrder - " & Err
End Function

```

```

'=====
'this function cancels the order if all the items in the order are cancelled
'args - order ID, DB level objects for order and items
'returns nothing
'=====

```

```

Private Function CancelOrder(ByVal alOrderID As Long, oDBord As DawnOrder.DBordHdr,
oDBItem As DawnOrder.DBordItem) As Boolean
On Error GoTo ErrHandler

```

```

Dim oRSorder As ADODB.Recordset
Dim oRSItems As ADODB.Recordset
Dim bItemNotCancelled As Boolean
Dim bOrderCancelled As Boolean
Dim lRow As Long

'get all the order and item records again
Set oRSorder = oDBord.GetOrder(alOrderID)
Set oRSItems = oDBItem.GetOrderItems(alOrderID)
bOrderCancelled = False
'check all the order item statuses for cancelled

```

```

                                BOrder.cls
    If Not ORSItems.EOF Then
        Do Until ORSItems.EOF
            If ORSItems("item_stus_cd") <>
DawnOrder.DawnItemStatus.osItemCancelled Then
                bItemNotCancelled = True
                Exit Do
            End If
            ORSItems.MoveNext
        Loop
    End If

    'boolean will be -ve if all items are cancelled, then set the orderstatus to
cancelled
    If Not bItemNotCancelled Then

        'update order status to cancelled
        lRow = oDBOrd.StopOrder(alOrderID,
DawnOrder.DawnOrderStatus.osCancelled, _
                                ORSOrder("last_maint_dtm"))

        'check error
        If lRow <> 1 Then
            RaiseResError
DawnOrder.DawnOrderErrors.ecUpdateOrderStatusFailed, _
                                TypeName(Me), "CancelOrder"
        End If
        bOrderCancelled = True

    End If

    Set ORSOrder = Nothing
    Set ORSItems = Nothing
    CancelOrder = bOrderCancelled
    Exit Function

```

ErrHandler:

```

    Set ORSOrder = Nothing
    Set ORSItems = Nothing
    RaiseError TypeName(Me), "CancelOrder"
    Exit Function

```

End Function

```

'''=====
''this function figures out what the From Email should be for Emails going to
customers based on what stage of the
''pipeline and order is, and drop ship indicators
''args - manfID, order id, item numbbber
''returns from Email as string
''=====
Private Function GetFromEmail(ByVal alManfID As Long, ByVal alOrderID As Long,
ByVal alItemNum As Long, _
                                oManf As DawnManf.UManf, oDBOrd As
DBOrdHdr, oDBItem As DBOrdItem) As String
    On Error GoTo ErrHandler
    Dim sEmail As String
    Dim ORSManf As ADODB.Recordset
    Dim ORSOrder As ADODB.Recordset
    Dim ORSItem As ADODB.Recordset
    Dim sDropShip As String

```

Border.cls

```

'
' Set oRsManf = oManf.GetManfByID(alManfID)
' Set oRsOrder = oDBOrd.GetOrder(alOrderID)
' Set oRsItem = oDBItem.GetOrderItem(alOrderID, alItemNum)
'
' sDropShip = IIf(IsNull(oRsOrder!drop_ship_ind), "0", oRsOrder!drop_ship_ind)
'
' If sDropShip = "0" Then
'     sEmail = IIf(IsNull(oRsManf!ord_email_addr), oRsManf!admin_email_addr,
oRsManf!ord_email_addr)
' Else
'     sEmail = GetFromEmail(oRsOrder!src_mfr_id, oRsOrder!pl_ord_id,
oRsItem!pl_item_num, oManf, oDBOrd, oDBItem)
' End If
'
' GetFromEmail = sEmail
'
' Set oRsManf = Nothing
' Set oRsOrder = Nothing
' Set oRsItem = Nothing
'
' Exit Function
'
'ErrorHandler:
'
' Set oRsManf = Nothing
' Set oRsOrder = Nothing
' Set oRsItem = Nothing
'
' RaiseError TypeName(Me), "GetFromEmail"
' Exit Function
'
'End Function

Private Function IntAddOrder( _
ByVal alUserID As Long, ByVal alOrdSrcManfID As Long, ByVal alOrgID As Long,
ByVal alManfID As Long, _
ByVal alShipFromManfID As Long, ByVal asPONumber As String, _
ByVal asRefNumber As String, ByVal asShipToName As String, _
ByVal asShipToAddr1 As String, ByVal asShipToAddr2 As String, _
ByVal asShipToCity As String, ByVal asShipToState As String, ByVal asShipToZip
As String, _
ByVal asShipToCountry As String, ByVal asBillToName As String, ByVal
asBillToAddr1 As String, _
ByVal asBillToAddr2 As String, ByVal asBillToCity As String, ByVal asBillToState
As String, _
ByVal asBillToZip As String, ByVal asBillToCountry As String, ByVal
asCreditcardNumber As String, _
ByVal asCreditCardExpiryDate As String, ByVal asDropShipIndicator As String, _
ByVal asBillTypeCD As String, ByVal asShipBriefDescr As String, ByVal acTax As
Currency, _
ByVal acShippingCost As Currency, ByVal asInternalNotes As String, _
ByVal asEmail As String, ByVal asIPAddress As String, ByVal asPipelineNotes As
String, _
ByVal asShippingInstructions As String, ByVal asShipToPhone As String, ByVal
asShipToExt As String, _
ByVal asBillToPhone As String, ByVal asBillToExt As String, ByVal
asBillToAltPhone As String, _
ByVal asBillToAltExt As String, ByVal asBillBriefDescr As String, _
ByVal asBillDescr As String, ByVal asShipDescr As String, _
ByVal asShipToCompanyNme As String, ByVal asBillToCompanyNme As String, _
ByVal alOnBehalfOfManfID As Long, ByVal asCreditcardName As String, ByVal asURL

```


Border.cls

```

As String, _
Optional ByVal oRSPipeItems As ADODB.Recordset) As ADODB.Recordset

On Error GoTo ErrHandler

Dim oContext As ObjectContext
Dim oManf As DawnManf.UManf
Dim oRSOrg As ADODB.Recordset
Dim oRSManf As ADODB.Recordset
Dim oRSManfUrl As ADODB.Recordset
Dim lOrderID As Long
Dim sURL As String
Dim oDBOrderHeader As DawnOrder.DBOrdHdr
Dim oSeq As DawnSeq.BSequence
Dim oDBManf As DawnManf.DBOrder
Dim oRSbasket As ADODB.Recordset
Dim oShop As DawnShop.DBShop
Dim oDBItem As DawnOrder.DBOrdItem
Dim oUser As DawnUser.UMember
Dim oRSUser As ADODB.Recordset
Dim lOrderNumber As Long
Dim lRow As Long
Dim lCounter As Long
Dim lNull As Long
Dim sSoldToFirstName As String
Dim sSoldToLastName As String
Dim sSoldToAddr1 As String
Dim sSoldToAddr2 As String
Dim sSoldToCity As String
Dim sSoldToState As String
Dim sSoldToZip As String
Dim sSoldToCountry As String
Dim lDesignID As Long
Dim sSoldToCompName As String
Dim sEmail As String
Dim lOrgID As Long
Dim oAddr As DawnAddr.UAddr
Dim lHistoryID As Long

'disable commit
Set oContext = GetObjectContext
If Not oContext Is Nothing Then
    oContext.DisableCommit
End If

'business rule
'if the order is from a KIOSK mfr, then the order should be placed with the
source mfr
'directly and not with the kiosk
'(ss_ind=1, pl_ind=1 and unattend_oper_ind=1) for a kiosk in the mfr table
Set oManf = CtxCreateObject("DawnManf.UManf")
Set oRSManf = GetSQLRecordset(" SELECT ss_ind, pl_mfr_ind, unattend_oper_ind " &
-
" FROM mfr WHERE mfr_id = " &
CStr(alManfID))
Set oRSManfUrl = oManf.GetManfUrls(alManfID)
If oRSManfUrl.RecordCount > 0 Then sURL = oRSManfUrl!base_url

If oRSManf("ss_ind") = 1 And oRSManf("pl_mfr_ind") = 0 And
oRSManf("unattend_oper_ind") = 1 Then
    'get the mfr_id for source mfr from the org table
    Set oRSOrg = GetSQLRecordset("SELECT org_id, mfr_id FROM org WHERE
pl_mfr_id = " & CStr(alManfID))

```

```

Border.cls
'set that to be the current mfr
If Not ORSOrg.EOF Then
    alOnBehalfOfManfID = alManfID
    alManfID = ORSOrg("mfr_id")
End If
Set ORSOrg = Nothing
End If

'generate order ID sequence
Set oSeq = CtxCreateObject("DawnSeq.BSequence")
lorderID = oSeq.GetNextInSequence("ord_hdr")
Set oSeq = Nothing

'get an order number from Manufacturer DB component and check for error
Set oDBManf = CtxCreateObject("DawnManf.DBOrder")
lorderNumber = oDBManf.GetNextOrderNumber(alManfID)
If IsNull(lorderNumber) Or lorderNumber <= 0 Then
    RaiseResError DawnOrderErrors.ecManfOrderNumberGenerationFailed,
    TypeName(Me), "IntAddOrder"
End If

'get the user address details for regular orders and mfr address for pipeline
orders
If alOrdSrcManfID <= 0 Then
    Set oUser = CtxCreateObject("DawnUser.UMember")
    Set oRSUser = oUser.GetMember(aluserID)
    ssoldToFirstName = oRSUser("frst_nme")
    ssoldToLastName = oRSUser("lst_nme")
    ssoldToAddr1 = oRSUser("addr_1")
    ssoldToAddr2 = IIf(IsNull(oRSUser("addr_2")), vbNullString,
oRSUser("addr_2"))
    ssoldToCity = oRSUser("city")
    ssoldToState = oRSUser("st_prov")
    ssoldToZip = oRSUser("zip_pstl_cd")
    ssoldToCountry = oRSUser("country")
    ssoldToCompName = oRSUser("company_nme")
    If Not IsNull(oRSUser("org_id")) Then lorgID = oRSUser("org_id")
Else
    Set oRSUser = oManf.GetManfByID(alOrdSrcManfID)
    ssoldToLastName = oRSUser("mfr_nme")
    Set oRSUser = Nothing
    Set oAddr = CtxCreateObject("DawnAddr.UAddr")
    Set oRSUser = oAddr.GetManfAddress(alOrdSrcManfID, 2)
    If oRSUser.RecordCount > 0 Then
        ssoldToFirstName = ""
        ssoldToAddr1 = oRSUser("addr_1")
        ssoldToAddr2 = IIf(IsNull(oRSUser("addr_2")), vbNullString,
oRSUser("addr_2"))
        ssoldToCity = oRSUser("city")
        ssoldToState = oRSUser("st_prov")
        ssoldToZip = oRSUser("zip_pstl_cd")
        ssoldToCountry = oRSUser("country")
        ssoldToCompName = ""
    End If
    lorgID = 0
End If

'call DB function
Set oDBOrderHeader = CtxCreateObject("DawnOrder.DBordHdr")
lRow = oDBOrderHeader.AddOrder( _
    lorderID, aluserID, alOrdSrcManfID, alorgID, alManfID, lorderNumber, _
    alShipFromManfID, Now, DawnOrderStatus.osSubmitted, asPONumber, asRefNumber,

```

```

        Border.cls
        sSoldToFirstName, sSoldToLastName, sSoldToAddr1, sSoldToAddr2, sSoldToCity,
-
        sSoldToState, sSoldToZip, sSoldToCountry, _
        asShipToName, asShipToAddr1, asShipToAddr2, asShipToCity, asShipToState,
asShipToZip, asShipToCountry, _
        asBillToName, asBillToAddr1, asBillToAddr2, asBillToCity, asBillToState,
asBillToZip, asBillToCountry, _
        asCreditCardNumber, asCreditCardExpiryDate, asDropShipIndicator,
asBillTypeCD, _
        asShipBriefDescr, acTax, acShippingCost, asInternalNotes, asPipelineNotes, _
        asShippingInstructions, asShipToPhone, asShipToExt, asBillToPhone,
asBillToExt, _
        asBillToAltPhone, asBillToAltExt, asBillBriefDescr, asBillDescr,
asShipDescr, _
        sSoldToCompName, asShipToCompanyNme, asBillToCompanyNme, alOnBehalfOfManfID,
asCreditCardName, asURL)

        'check for error
        If lRow <> 1 Then
            RaiseResError DawnOrderErrors.ecAddOrderFailed, TypeName(Me), "IntAddOrder"
        End If

        'add row to order history table
        Set oDBOrdHist = CtxCreateObject("DawnOrder.DBOrdHist")
        lHistoryID = AddOrderHistory(lOrderID, 1, aluserID, "", asEmail, 0, asIPAddress,
DawnOrderStatus.osSubmitted)

        Set oDBItem = CtxCreateObject("DawnOrder.DBOrdItem")
        Set oDBItemHist = CtxCreateObject("DawnOrder.DBOrdItemHist")

        'If not a pipelined order
        If alOrdSrcManfID <= 0 Then
            'add all shopping basket items to the order items table
            Set oShop = CtxCreateObject("DawnShop.DBShop")
            Set oRSbasket = oShop.GetItems(aluserID)

            'raise an error if there are no items in the basket
            If oRSbasket.EOF Then
                RaiseResError DawnOrder.DawnOrderErrors.ecNoItemsInBasket,
TypeName(Me), "IntAddOrder"
            End If

            'create db comp for order items and add all the items and also make
an entry into the item history table
            lCounter = 1
            oRSbasket.MoveFirst
            Do Until oRSbasket.EOF
                'add order item
                lDesignID = 0 'clears any previous value
                If Not IsNull(oRSbasket("design_id")) Then
                    lDesignID = oRSbasket("design_id")
                End If
                lRow = oDBItem.AddOrderItem( _
                    lOrderID, lCounter,
oRSbasket("part_id"), lNull, lNull, _
                    CStr(DawnItemStatus.osItemSubmitted),
oRSbasket("qty"), lNull, lNull, _
                    oRSbasket("unit_price"), lDesignID)

                'check for errors
                If lRow <> 1 Then
                    RaiseResError DawnOrderErrors.ecAddOrderItemFailed,
TypeName(Me), "IntAddOrder"
                End If
            End If
        End If
    End Sub

```

```

                                BOrder.cls

                                'add order item history
                                AddOrderItemHistory lOrderID, lCounter, "New Item Added",
asEmail, asIPAddress, osItemSubmitted

                                lCounter = lCounter + 1
                                oRSbasket.MoveNext

                                Loop

                                'delete all the shopping items from the shopping basket
                                oRSbasket.MoveFirst
                                Do Until oRSbasket.EOF
                                    lRow = oShop.DeleteItemFromBasket( _
                                        aluserID, oRSbasket("item_seq_num"),
oRSbasket("last_maint_dtm"))

                                    'check for errors
                                    If lRow <> 1 Then
                                        RaiseResError
DawnOrderErrors.ecDeleteShoppingBasketItemFailed, TypeName(Me), "IntAddOrder"
                                    End If
                                    oRSbasket.MoveNext

                                Loop

                                'Send e-mail to user.
                                UpdateOrderNotes lOrderID, lHistoryID, asInternalNotes,
asPipelineNotes, "", asShippingInstructions
                                sEmail = CreateEmail(lOrderID, asEmail, emAddOrder)

                                Else 'Pipeline order - notes are updated in Pipeline function for pipeline
orders

                                    lDesignID = 0 'Clears any previous value
                                    lCounter = 1
                                    If Not oRSPipeItems.EOF Then oRSPipeItems.MoveFirst
                                    Do While Not oRSPipeItems.EOF

                                        If Not IsNull(oRSPipeItems("design_id")) Then
                                            lDesignID = oRSPipeItems("design_id")
                                        End If
                                        lRow = oDBItem.AddOrderItem( _
                                            lOrderID, lCounter,
oRSPipeItems("part_id"), oRSPipeItems!pipe_ord_id, _
                                            oRSPipeItems!pipe_item_num,
CStr(DawnItemStatus.osItemSubmitted), _
                                            oRSPipeItems!ord_qty, lNull, lNull,
oRSPipeItems!unit_price, lDesignID)
                                        'check for errors
                                        If lRow <> 1 Then
                                            RaiseResError DawnOrderErrors.ecAddOrderItemFailed,
TypeName(Me), "IntAddOrder"
                                        End If

                                        'add order item history
                                        AddOrderItemHistory lOrderID, lCounter, "New Item Added",
asEmail, asIPAddress, osItemSubmitted

                                        oRSPipeItems.MoveNext
                                        lCounter = lCounter + 1

                                    Loop

                                    'send email to src mfr

```

```

                                Border.cls
                                UpdateOrderNotes lOrderID, lHistoryID, "", "", "",
asShippingInstructions
                                sEmail = CreateEmail(lOrderID, asEmail, emAddOrder, , , , , True)

End If

'Send e-amil to mfr for regular orders and pipeline mfr for pipeline orders
If alOrdSrcManfID <= 0 Then
    sEmail = CreateEmail(lOrderID, "", emAddOrder, , , , lOrgID)
Else
    sEmail = CreateEmail(lOrderID, "", emAddOrder, , , , lOrgID, True)
End If

'set up default src mfrs for the items if the items have only one source mfr
Call SetSourceMfr(alManfID, lOrderID)

Set IntAddOrder = oDBOrderHeader.GetOrder(lOrderID)

Set oRsManf = Nothing
Set oRSManfUrl = Nothing
Set oManf = Nothing
Set oRSOrg = Nothing
Set oContext = Nothing
Set oDBOrderHeader = Nothing
Set oSeq = Nothing
Set oDBManf = Nothing
Set oRSbasket = Nothing
Set oShop = Nothing
Set oDBItem = Nothing
'    Set oDBOrdHist = Nothing
'    Set oDBItemHist = Nothing
Set oUser = Nothing
Set oRSUser = Nothing
Set oAddr = Nothing

If Not oContext Is Nothing Then
    oContext.EnableCommit
End If
CtxSetComplete

Exit Function

'error
ErrHandler:
If Not oContext Is Nothing Then
    oContext.EnableCommit
End If
CtxSetAbort

Set oRsManf = Nothing
Set oRSManfUrl = Nothing
Set oManf = Nothing
Set oRSOrg = Nothing
Set oContext = Nothing
Set oDBOrderHeader = Nothing
Set oSeq = Nothing
Set oDBManf = Nothing
Set oRSbasket = Nothing
Set oShop = Nothing
Set oDBItem = Nothing
'    Set oDBOrdHist = Nothing
'    Set oDBItemHist = Nothing
Set oUser = Nothing

```

BOrder.cls

```

Set ORSUser = Nothing
Set oAddr = Nothing

RaiseError TypeName(Me), "IntAddOrder"
Exit Function

End Function

'=====
'this function updates an order
'input - all the fields in the ORD_HDR table - except status - status is not updated
'here
'record set with the updated order
'=====

Public Function UpdateOrder(ByVal alOrderID As Long, _
    ByVal asPONumber As String, ByVal asRefNumber As String, _
    ByVal asShipToName As String, ByVal asShipToAddr1 As String, ByVal asShipToAddr2
As String, _
    ByVal asShipToCity As String, ByVal asShipToState As String, ByVal asShipToZip
As String, _
    ByVal asShipToCountry As String, ByVal asBillToName As String, ByVal
asBillToAddr1 As String, _
    ByVal asBillToAddr2 As String, ByVal asBillToCity As String, ByVal asBillToState
As String, _
    ByVal asBillToZip As String, ByVal asBillToCountry As String, _
    ByVal asCreditcardNumber As String, ByVal asCreditCardExpiryDate As String, _
    ByVal asDropShipIndicator As String, ByVal alBillTypeCD As String, ByVal
alShipBriefDescr As String, _
    ByVal acTax As Currency, ByVal acShippingCost As Currency, ByVal asInternalNotes
As String, _
    ByVal adMaintDate As Date, ByVal alEmailInd As Long, ByVal asEmailNotes As
String, _
    ByVal asEmail As String, ByVal asIPAddress As String, asPipelineNotes As String,
_
    ByVal asShippingInstructions As String, ByVal ShipToPhone As String, ByVal
ShipToExt As String, _
    ByVal BillToPhone As String, ByVal BillToExt As String, ByVal BillToAltPhone As
String, _
    ByVal BillToAltExt As String, ByVal asCreditcardName As String, ByVal aluserID
As Long, _
    ByVal alBillBriefDescr As String, ByVal alBillDescr As String, ByVal alShipDescr
As String, _
    ByVal asSoldToCompanyNme As String, ByVal asShipToCompanyNme As String, _
    ByVal asBillToCompanyNme As String) As ADODB.Recordset

    On Error GoTo ErrHandler

    'var
    Dim oDBOrderHeader As DawnOrder.DBOrdHdr
    ' Dim oDBhist As DawnOrder.DBOrdHist
    Dim lRow As Long
    Dim lNull As Long
    Dim oRS As ADODB.Recordset
    Dim lHistoryID As Long

    'call DB level function
    Set oDBOrderHeader = CtxCreateObject("DawnOrder.DBOrdHdr")
    lRow = oDBOrderHeader.UpdateOrder( _
        alOrderID, asPONumber, asRefNumber, _
        asShipToName, asShipToAddr1, asShipToAddr2, asShipToCity, asShipToState,
asShipToZip, asShipToCountry, _
        asBillToName, asBillToAddr1, asBillToAddr2, asBillToCity, asBillToState,

```

Border.cls

```

asBillToZip, asBillToCountry, _
    asCreditcardNumber, asCreditCardExpiryDate, asDropShipIndicator,
alBillTypeCD, alShipBriefDescr, _
    acTax, acShippingCost, asInternalNotes, adMaintDate, asPipelineNotes,
asShippingInstructions, _
    alEmailInd, ShipToPhone, ShipToExt, BillToPhone, BillToExt, BillToAltPhone,
BillToAltExt, _
    asCreditcardName, aluserID, alBillBriefDescr, alBillDescr, alShipDescr,
asSoldToCompanyNme, _
    asShipToCompanyNme, asBillToCompanyNme)

'check for errors
If lRow <> 1 Then
    RaiseResError DawnOrder.DawnOrderErrors.ecUpdateOrderFailed, TypeName(Me),
"UpdateOrder"
End If

Set oRS = oDBOrderHeader.GetOrder(alOrderID)

'call Db function to add order history and notes
Set oDBhist = CtxCreateObject("DawnOrder.DBOrdHist")
lHistoryID = AddOrderHistory(alOrderID, alEmailInd, aluserID, "", asEmail, 0,
asIPAddress, oRS!ord_stus_cd)
UpdateOrderNotes alOrderID, lHistoryID, asInternalNotes, asPipelineNotes,
asEmailNotes

'get record set of the updated order
Set UpdateOrder = oRS

'end function
CtxSetComplete
Set oDBOrderHeader = Nothing
Set oDBhist = Nothing
Exit Function

'error
ErrHandler:
CtxSetAbort
Set oDBOrderHeader = Nothing
Set oDBhist = Nothing
RaiseError TypeName(Me), "UpdateOrder"

End Function

'=====
'this function updates an order item
'input - all the fields in the ORD_item table - except status - status is not
updated here
'record set with the updated order item
'=====
Public Function UpdateOrderItem(
    ByVal alOrderID As Long, ByVal alItemNumber As Long, ByVal alPartID As Long, _
    ByVal alPipelineOrderID As Long, ByVal alPipelineItemNumber As Long, _
    ByVal alOrderQty As Long, ByVal alShipQty As Long, _
    ByVal alReturnQty As Long, ByVal acUnitPrice As Currency, ByVal alDesignID As
Long, _
    ByVal adMaintDate As Date, ByVal asInternalNotes As String, ByVal alEmailInd As
Long, _
    ByVal asEmailNotes As String, ByVal asEmail As String, ByVal asIPAddress As
String, _
    ByVal asPipelineNotes As String, ByVal asShippingInstructions As String, ByVal
aluserID As Long) _
    As ADODB.Recordset

```

BOrder.cls

```

On Error GoTo ErrHandler

'var
Dim oDBObject As DawnOrder.DBordItem
' Dim oDBItemHist As DawnOrder.DBordItemHist
' Dim oDBObject As DawnOrder.DBordHdr
' Dim oDBordHist As DawnOrder.DBordHist
Dim oRS As ADODB.Recordset
Dim lRow As Long
Dim lHistoryID As Long

'update order item - call db function
Set oDBObject = CtxCreateObject("DawnOrder.DBordItem")
lRow = oDBObject.UpdateOrderItem( _
    alOrderID, alItemNumber, alPartID, alPipelineOrderID,
alPipelineItemNumber, _
    alOrderQty, alShipQty, alReturnQty, acUnitPrice, alDesignID,
adMaintDate)

'check for errors
If lRow <> 1 Then
    RaiseResError DawnOrder.DawnOrderErrors.ecUpdateOrderItemFailed, _
        TypeName(Me), "UpdateOrderItem"
End If

Set oRS = oDBObject.GetOrderItem(alOrderID, alItemNumber)

'enter a row in item history table
' Set oDBItemHist = CtxCreateObject("DawnOrder.DBordItemHist")
AddOrderItemHistory alOrderID, alItemNumber, "Order Item was Updated", asEmail,
asIPAddress, oRS!item_stus_cd

'update order history
Set oDBObject = CtxCreateObject("DawnOrder.DBordHdr")
Set oRS = oDBObject.GetOrder(alOrderID)
' Set oDBordHist = CtxCreateObject("dawnOrder.DBordHist")
lHistoryID = AddOrderHistory(alOrderID, alEmailInd, aluserID, asEmailNotes,
asEmail, 0, asIPAddress, oRS!ord_stus_cd)
UpdateOrderNotes alOrderID, lHistoryID, asInternalNotes, asPipelineNotes,
asEmailNotes

'get record set of the updated order
Set UpdateOrderItem = oDBObject.GetOrderItem(alOrderID, alItemNumber)

'exit function
CtxSetComplete
Set oDBObject = Nothing
' Set oDBItemHist = Nothing
Set oDBObject = Nothing
' Set oDBordHist = Nothing
Exit Function

'error
ErrHandler:
CtxSetAbort
Set oDBObject = Nothing
' Set oDBItemHist = Nothing
Set oDBObject = Nothing
' Set oDBordHist = Nothing
RaiseError TypeName(Me), "UpdateOrderItem"

End Function

```


Border.cls

```

'=====
'=====
' this function updates the order item mfr source for the item ,
' input - Order ID, 2D array with item number and source manf id
' for 2 nd dimension : 0 element is item number, 1 element is source mfr ID
' none
'=====
Public Sub UpdateOrderItemSource(ByVal aOrderID As Long, ByVal arrItemManfArray As
Variant, _
ByVal asInternalNotes As String, ByVal adOrderMaintDate As Date, ByVal asEmail
As String, _
ByVal asIPAddress As String, ByVal asEmailNotes As String, ByVal alEmailInd As
Long, _
ByVal asPipelineNotes As String, ByVal asShippingInstructions As String, ByVal
aluserID As Long)

On Error GoTo ErrHandler

'vars
Dim oDBOrder As DawnOrder.DBOrdHdr
Dim oDBOrdHist As DawnOrder.DBOrdHist
Dim oDBItem As DawnOrder.DBOrdItem
Dim oDBItemHist As DawnOrder.DBOrdItemHist
Dim lItemNumber As Long
Dim lManfID As Long
Dim lRow As Long
Dim lCounter As Long
Dim lStatus As Long
Dim lStatusOriginal As Long
Dim oRSItem As ADODB.Recordset
Dim lHistoryID As Long

'create DB component to add the data
Set oDBOrder = CtxCreateObject("DawnOrder.DBOrdHdr")
Set oDBOrdHist = CtxCreateObject("DawnOrder.DBOrdHist")
Set oDBItem = CtxCreateObject("DawnOrder.DBOrdItem")
Set oDBItemHist = CtxCreateObject("DawnOrder.DBOrdItemHist")

'loop thru the array to get the item numbers and source mfr IDs
For lCounter = 0 To UBound(arrItemManfArray)

'get item number and source mfr ID from array
lItemNumber = arrItemManfArray(lCounter, 0)
lManfID = arrItemManfArray(lCounter, 1)

'change only if value >= 0
If lManfID >= 0 Then
'check to make sure item has valid status for source to be updated
'item can be submitted, to Mfg, to Pipeline, pipeline reject or
pipeline cancelled only
Set oRSItem = oDBItem.GetOrderItem(aOrderID, lItemNumber)
If oRSItem.EOF Then
RaiseResError
DawnOrder.DawnOrderErrors.ecOrderItemNumberNotValid, TypeName(Me), _
"UpdateOrderItemSource(" & "Order ID: " & CStr(aOrderID) &
", Item Number : " & CStr(lItemNumber) & ")"
End If
lStatusOriginal = CLng(oRSItem!item_stus_cd)
If Not (lStatusOriginal = DawnOrder.DawnItemStatus.osItemSubmitted
Or _
lStatusOriginal = DawnOrder.DawnItemStatus.osItemToPipeline Or _

```

```

Border.cls
    lStatusOriginal = DawnOrder.DawnItemStatus.osItemReleasedtoManf
Or _
    lStatusOriginal =
DawnOrder.DawnItemStatus.osItemPipelineCancelled Or _
    lStatusOriginal =
DawnOrder.DawnItemStatus.osItemPipelineRejected) Then
    RaiseResError
DawnOrder.DawnOrderErrors.ecItemSourceUpdateNotAllowed, TypeName(Me), _
    "UpdateOrderItemSource(" & "Order ID: " & CStr(alOrderID) &
    ", Item Number : " & CStr(lItemNumber) & ")"
    End If
    lStatus = IIf(lManfID = 0, DawnItemStatus.osItemReleasedtoManf,
DawnItemStatus.osItemToPipeline)

    'update source mfr to item
    lRow = ODBItem.UpdateOrderItemSource(alOrderID, lItemNumber,
lManfID)
    If lRow <> 1 Then
        RaiseResError DawnOrder.DawnOrderErrors.ecUpdateOrderItemFailed,
        TypeName(Me), "UpdateOrderItemSource"
    End If

    'update item status if item is not submitted
    If Not lStatusOriginal = DawnOrder.DawnItemStatus.osItemSubmitted
Then
        lRow = ODBItem.UpdateOrderItemStatus(alOrderID, lItemNumber,
lStatus)
        If lRow <> 1 Then
            RaiseResError
DawnOrder.DawnOrderErrors.ecUpdateOrderItemStatusFailed, _
            TypeName(Me), "UpdateOrderItemSource"
        End If
    End If

    'add row in history for item
    AddOrderItemHistory alOrderID, lItemNumber, _
        "Order Item Source Manufacturer and
Order Item Status were Updated", _
        asEmail, asIPAddress, lStatus
    End If
Next

'call Db function to add order history and notes
lHistoryID = AddOrderHistory(alOrderID, alEmailInd, aluserID, "", asEmail, 0,
asIPAddress, DawnOrderStatus.osInProgress)
UpdateOrderNotes alOrderID, lHistoryID, asInternalNotes, asPipelineNotes,
asEmailNotes

'end
CtxSetComplete

Set ODBOrder = Nothing
Set ODBordHist = Nothing
Set ODBItem = Nothing
Set ODBItemHist = Nothing

Exit Sub

ErrorHandler:
CtxSetAbort

```

BOrder.cls

```

    Set oDBOrder = Nothing
    Set oDBordHist = Nothing
    Set oDBItem = Nothing
    Set oDBItemHist = Nothing

    RaiseError TypeName(Me), "UpdateOrderItemSource"
End Sub

'=====
'this function updates an order status
'input - order id, status, last maint date, email, ip address
'record set with order details
'=====
Public Sub UpdateOrderStatus( _
    ByVal alOrderID As Long, ByVal alOrderStatusCode As Long, ByVal asInternalNotes
As String, _
    ByVal adMaintDate As Date, ByVal asEmail As String, ByVal asIPAddress As String,
    _
    ByVal asEmailNotes As String, ByVal alEmailIndicator As Long, _
    ByVal asPipelineNotes As String, ByVal asShippingInstructions As String, ByVal
aluserID As Long)

    On Error GoTo ErrHandler

    Call IntUpdateOrderStatus(alOrderID, alOrderStatusCode, asInternalNotes, _
        adMaintDate, asEmail, asIPAddress, asEmailNotes,
alEmailIndicator, asPipelineNotes, _
        asShippingInstructions, aluserID)

    SendEmails
    CtxSetComplete

    Exit Sub

'error
ErrHandler:
    CtxSetAbort
    RaiseError TypeName(Me), "UpdateOrderStatus"
End Sub

Private Sub IntUpdateOrderStatus( _
    ByVal alOrderID As Long, ByVal alOrderStatusCode As Long, ByVal asInternalNotes
As String, _
    ByVal adMaintDate As Date, ByVal asEmail As String, ByVal asIPAddress As String,
    _
    ByVal asEmailNotes As String, ByVal alEmailInd As Long, ByVal asPipelineNotes As
String, _
    ByVal asShippingInstructions As String, ByVal aluserID As Long)

    On Error GoTo ErrHandler

    'var
    Dim oDBOrder As DawnOrder.DBordHdr
    Dim oDBhist As DawnOrder.DBordHist
    Dim oDBItem As DawnOrder.DBordItem
    Dim oDBItemHist As DawnOrder.DBordItemHist
    Dim lRow As Long
    Dim lNull As Long
    Dim lOldOrderStatus As Long
    Dim oRS As ADODB.Recordset
    Dim oRSItems As ADODB.Recordset

```

```

                                Border.cls
Dim sItemHistoryDescription As String
Dim lItemStatus As Long
Dim lSrcMfr As Long
Dim lHistoryID As Long

'rule - order status cannot be set to partiallyShipped or shipped directly
'it will be done by the system, when item status is changed to shipped
If alOrderStatusCode = DawnOrder.DawnOrderStatus.osPartiallyShipped Or _
    alOrderStatusCode = DawnOrder.DawnOrderStatus.osShipped Then
    RaiseResError DawnOrder.DawnOrderErrors.ecOrderStatusNotAllowed,
TypeName(Me), "IntUpdateOrderStatus"
End If

'create DB objects
Set oDBOrder = CtxCreateObject("DawnOrder.DBOrdHdr")
Set oDBhist = CtxCreateObject("DawnOrder.DBOrdHist")

'create db objects for items
Set oDBItem = CtxCreateObject("DawnOrder.DBOrdItem")
Set oDBItemHist = CtxCreateObject("DawnOrder.DBOrdItemHist")

'rule - if order is partially shipped, fully shipped, cancelled, rejected,
onhold
'then no order status change is allowed
lOldOrderStatus = oDBOrder.GetOrderStatus(alOrderID)
If lOldOrderStatus = DawnOrder.DawnOrderStatus.osPartiallyShipped Or _
    lOldOrderStatus = DawnOrder.DawnOrderStatus.osShipped Then
    RaiseResError DawnOrder.DawnOrderErrors.ecOrderAlreadyShipped, TypeName(Me),
"IntUpdateOrderStatus"
End If
If lOldOrderStatus = DawnOrder.DawnOrderStatus.osCancelled Or _
    lOldOrderStatus = DawnOrder.DawnOrderStatus.osRejected Or _
    lOldOrderStatus = DawnOrder.DawnOrderStatus.osOnhold Then
    RaiseResError DawnOrder.DawnOrderErrors.ecOrderStopped, TypeName(Me),
"IntUpdateOrderStatus"
End If

'get order items to a record set
Set oRSItems = oDBItem.GetOrderItems(alOrderID)

'7/13/2000 uday
'rule - if there is any item in pipeline processing then, if the mfr is trying to
PUT BACK
'an order, do not allow the order status change
If (lOldOrderStatus = DawnOrder.DawnOrderStatus.osInProgress And
alOrderStatusCode = DawnOrder.DawnOrderStatus.osCreditReview) Or _
    (lOldOrderStatus = DawnOrder.DawnOrderStatus.osCreditReview And
alOrderStatusCode = DawnOrder.DawnOrderStatus.osSubmitted) Then
    'if there is even one item with pipeline status, do not allow this
change
    If oRSItems.RecordCount > 0 Then oRSItems.MoveFirst
    Do While Not oRSItems.EOF
        If oRSItems!item_stus_cd =
DawnOrder.DawnItemStatus.osItemInPipeline Or _
            oRSItems!item_stus_cd =
DawnOrder.DawnItemStatus.osItemPipelineCancelled Or _
            oRSItems!item_stus_cd =
DawnOrder.DawnItemStatus.osItemPipelineComplete Or _
            oRSItems!item_stus_cd =
DawnOrder.DawnItemStatus.osItemPipelineDropShipped Or _
            oRSItems!item_stus_cd =
DawnOrder.DawnItemStatus.osItemPipelineRejected Then
            RaiseResError ecOrderStatusNotAllowed,
            Page 16
        End If
    Loop
End If

```

```

Border.cls
TypeName(Me), "IntUpdateOrderStatus"
    End If
    ORSItems.MoveNext
Loop
End If

ORSItems.MoveFirst
'call DB level function - update order status
lRow = oDBOrder.UpdateOrderStatus(alOrderID, CStr(alOrderStatusCode),
adMaintDate)

'check for errors
If lRow <> 1 Then
    RaiseResError DawnOrder.DawnOrderErrors.ecUpdateOrderStatusFailed,
TypeName(Me), "IntUpdateOrderStatus"
    & CStr(alOrderID) & "-" & CStr(alOrderStatusCode) & " " &
CStr(adMaintDate)
End If

'call Db function to add order history
lHistoryID = AddOrderHistory(alOrderID, alEmailInd, aluserID, "", asEmail,
lNull, asIPAddress, alOrderStatusCode)

If alOrderStatusCode = DawnOrder.DawnOrderStatus.osSubmitted Or _
alOrderStatusCode = DawnOrder.DawnOrderStatus.osCreditReview Or _
alOrderStatusCode = DawnOrder.DawnOrderStatus.osInProgress Then

    If Not ORSItems.EOF Then
        Do Until ORSItems.EOF
            'get item source mfr, based on source, item will either go to mfg
que or pipeline que
            'new rule, if no src mfr, then set to pipeline 6/13/00 uday
            If Not IsNull(ORSItems!src_mfr_id) Then
                lSrcMfr = ORSItems!src_mfr_id
            Else
                lSrcMfr = -1
            End If
            Select Case alOrderStatusCode
                Case Is = DawnOrder.DawnOrderStatus.osInProgress
                    'set up item status code based on src mfr Id
                    If lSrcMfr <> 0 Then
                        lItemStatus = DawnOrder.DawnItemStatus.osItemToPipeline
                        sItemHistoryDescription = "Order Item Status was Updated
to To Pipeline"
                    Else
                        lItemStatus =
DawnOrder.DawnItemStatus.osItemReleasedtoManf
                        sItemHistoryDescription = "Order Item Status was Updated
to Released To Manufacturer"
                    End If
                Case Is = DawnOrder.DawnOrderStatus.osCreditReview
                    sItemHistoryDescription = "Order Item Status was Updated to
Item Submitted"
                Case Is = DawnOrder.DawnItemStatus.osItemSubmitted
                    lItemStatus = DawnOrder.DawnItemStatus.osItemSubmitted
                Case Is = DawnOrder.DawnOrderStatus.osSubmitted
                    sItemHistoryDescription = "Order Item Status was Updated to
Item Submitted"
                lItemStatus = DawnOrder.DawnItemStatus.osItemSubmitted
            End Select
            'Update order item status
            lRow = oDBItem.UpdateOrderItemStatus(alOrderID,
ORSItems("item_num"), _

```

```

                                BOrder.cls
                                CStr(lItemStatus))

                                'check for errors
                                If lRow <> 1 Then
                                    RaiseResError
                                DawnOrder.DawnOrderErrors.ecUpdateOrderItemStatusFailed, TypeName(Me),
                                "IntUpdateOrderStatus"
                                End If

                                'add item history
                                AddOrderItemHistory alOrderID, oRSItems("item_num"),
                                sItemHistoryDescription, asEmail, asIPAddress, lItemStatus

                                oRSItems.MoveNext
                                Loop
                            End If

                        End If

                        'call DB level function - update order status
                        UpdateOrderNotes alOrderID, lHistoryID, asInternalNotes, asPipelineNotes,
                        asEmailNotes

                        If alEmailInd = 1 Then
                            asEmailNotes = CreateEmail(alOrderID, asEmail, emUpdateOrderStatus,
                            asEmailNotes)
                        End If

                        'end function
                        CtxSetComplete
                        Set oDBOrder = Nothing
                        ' Set oDBhist = Nothing
                        Set oDBItem = Nothing
                        ' Set oDBItemHist = Nothing
                        Exit Sub

                        'error
                        ErrHandler:
                        CtxSetAbort
                        Set oDBOrder = Nothing
                        ' Set oDBhist = Nothing
                        Set oDBItem = Nothing
                        ' Set oDBItemHist = Nothing

                        RaiseError TypeName(Me), "IntUpdateOrderStatus"
                    End Sub
                    '=====
                    'this function updates an order item status
                    'input - order id, status, last maint date, email, ip address
                    ' record set with the updated order item detail
                    '=====
                    Public Function UpdateOrderItemStatus( _
                        ByVal alOrderID As Long, ByVal arrItemNumber As Variant, ByVal arrItemSource As
                        Variant, _
                        ByVal arrShipDate As Variant, ByVal arrTrackNum As Variant, ByVal
                        alOrderItemStatusCode As Long, _
                        ByVal asInternalNotes As String, ByVal adOrderMaintDate As Date, ByVal asEmail
                        As String, _
                        ByVal asIPAddress As String, ByVal asEmailNotes As String, ByVal alEmailInd As
                        Long, _
                        ByVal asPipelineNotes As String, ByVal asShippingInstructions As String, ByVal

```

BOrder.cls

```

aluserID As Long) _
    As ADODB.Recordset

    On Error GoTo ErrHandler

    Set UpdateOrderItemStatus = IntUpdateOrderItemStatus( _
        alOrderID, arrItemNumber, arrItemSource, arrShipDate, arrTrackNum,
alOrderItemStatusCode, asInternalNotes, _
        adOrderMaintDate, asEmail, asIPAddress, asEmailNotes, alEmailInd,
asPipelineNotes, asShippingInstructions, aluserID)

    SendEmails
    'exit
    CtxSetComplete
    Exit Function

    'error
ErrHandler:
    CtxSetAbort
    RaiseError TypeName(Me), "UpdateOrderItemStatus"

End Function

Private Function IntUpdateOrderItemStatus( _
    ByVal alOrderID As Long, ByVal arrItemNumber As Variant, ByVal arrItemSource As
Variant, _
    ByVal arrShipDate As Variant, ByVal arrTrackNum As Variant, ByVal
alOrderItemStatusCode As Long, _
    ByVal asInternalNotes As String, ByVal adOrderMaintDate As Date, ByVal asEmail
As String, _
    ByVal asIPAddress As String, ByVal asEmailNotes As String, ByVal alEmailInd As
Long, _
    ByVal asPipelineNotes As String, ByVal asShippingInstructions As String, ByVal
aluserID As Long, _
    Optional ByVal alProcess As Long = 1, Optional ByVal abFromPipeline As Boolean =
False) As ADODB.Recordset

    On Error GoTo ErrHandler

    'var
    Dim ODBOrderItem As DawnOrder.DBOrdItem
    Dim ODBItemHist As DawnOrder.DBOrdItemHist
    Dim ODBOrder As DawnOrder.DBOrdHdr
    Dim ODBOrdHist As DawnOrder.DBOrdHist
    Dim lRow As Long
    Dim lNull As Long
    Dim lOrderStatus As Long
    Dim oRSorder As ADODB.Recordset
    Dim oRSItems As ADODB.Recordset
    Dim bItemNotCancelled As Boolean
    Dim bItemNotShipped As Boolean
    Dim lNewOrderStatus As Long
    Dim lItemNumber As Long
    Dim lCounter As Long
    Dim lOldItemStatus As Long
    Dim dShipdate As Date
    Dim sTrackNum As String
    Dim bSource As Boolean
    Dim lItemStatus As Long
    Dim sBody As String
    Dim lOrderItemStatusCode As Long
    Dim oManf As DawnManf.UManf

```

Border.cls

```

Dim oRsManf As ADODB.Recordset
Dim sSQL As String
Dim oRSProcess As ADODB.Recordset
Dim bPipeline As Boolean
Dim sEmailNotes As String
Dim dTotal As Double
Dim lHistoryID As Long
Dim sEmailOriginal As String

'record set for handling items in sequence, so that orders are updated together
instead of one item at a time
Dim oRSUp As ADODB.Recordset
Dim oRSDn As ADODB.Recordset
Dim arrItems() As Long
Dim arrDate() As Date
Dim arrTrk() As String
Dim lOldID As Long
Dim lUbound As Long

'rule - item status cannot be changed to ItemSubmitted - this happens only
'when a new order is created
If alOrderItemStatusCode = DawnOrder.DawnItemStatus.osItemSubmitted Then
    RaiseResError DawnOrder.DawnOrderErrors.ecItemStatusNotAllowed,
        TypeName(Me), _
            "IntUpdateOrderItemStatus"
End If

' checks if arrItemSource exists.
' we need the on error resume next since arrItemSource is optional.
On Error Resume Next
    If UBound(arrItemNumber) = UBound(arrItemSource) Then
        If Err = 0 Then bSource = True
        Err.Clear
    End If
On Error GoTo ErrHandler

'create db objects
Set oDBOrderItem = CtxCreateObject("DawnOrder.DBOrdItem")
Set oDBItemHist = CtxCreateObject("DawnOrder.DBOrdItemHist")
Set oDBOrder = CtxCreateObject("DawnOrder.DBOrdHdr")
Set oDBOrdHist = CtxCreateObject("DawnOrder.DBOrdHist")
Set oManf = CtxCreateObject("DawnManf.UManf")

'rule - if order status is OnHold, Rejected or Cancelled then
'item statuses cannot be changed

'get order status
Set oRSorder = oDBOrder.GetOrder(alOrderID)
lOrderStatus = oRSorder("ord_stus_cd")

'check order status, if order is cancelled, change of status is allowed
if item is in pipeline
    If lOrderStatus = DawnOrder.DawnOrderStatus.osRejected Or _
        lOrderStatus = DawnOrder.DawnOrderStatus.osOnhold Then
        RaiseResError
        DawnOrder.DawnOrderErrors.ecItemStatusChangeNotAllowed, _
            TypeName(Me), "IntUpdateOrderItemStatus"
    End If

'end rule check
'check for pipeline order
bPipeline = False
sEmailOriginal = asEmailNotes

```


Border.cls

```

'set up pipeline record sets
Set oRSUp = GetClientRS()
oRSUp.Fields.Append "ord_id", adInteger
oRSUp.Fields.Append "item_num", adInteger
oRSUp.Fields.Append "ship_dte", adDate
oRSUp.Fields.Append "trk_num", adVarChar, 255
oRSUp.Fields.Append "item_stus", adInteger
oRSUp.Open
Set oRSDn = GetClientRS()
oRSDn.Fields.Append "ord_id", adInteger
oRSDn.Fields.Append "item_num", adInteger
oRSDn.Open

'***** in the loop for variant
For lCounter = 0 To UBound(arrItemNumber) Step 1

    'get item number
    lItemNumber = CLng(arrItemNumber(lCounter))

    On Error Resume Next
    dShipdate = CDate(arrShipDate(lCounter))
    sTrackNum = CStr(arrTrackNum(lCounter))
    Err.Clear

    'resets error handler
    On Error GoTo ErrHandler

    'get old item status
    lOldItemStatus = oDBOrderItem.GetOrderItemStatus(alOrderID, lItemNumber)

    'if item is cancelled, then do not change item status
    If lOldItemStatus = DawnOrder.DawnItemStatus.osItemCancelled Then
        RaiseResError
        DawnOrder.DawnOrderErrors.ecItemStatusChangeNotAllowed, _
            TypeName(Me), "IntUpdateOrderItemStatus"
    End If

    'if order is cancelled, then only item status with pipelinecancelled
    status can be changed, and the order status
    'will be realised later in the code
    If lOrderStatus = DawnOrder.DawnOrderStatus.osCancelled And _
        lOldItemStatus <> DawnOrder.DawnItemStatus.osItemPipelineCancelled
    Then
        RaiseResError
        DawnOrder.DawnOrderErrors.ecItemStatusChangeNotAllowed, _
            TypeName(Me), "IntUpdateOrderItemStatus"
    End If

    'The 'In's status (ReleasedToManf and ToPipeline) depend on the source.
    'This checks the source and adjust the status.
    lItemStatus = alOrderItemStatusCode
    If alOrderItemStatusCode = DawnOrder.DawnItemStatus.osItemReleasedtoManf
    Or _
    Then
        alOrderItemStatusCode = DawnOrder.DawnItemStatus.osItemToPipeline
        If bSource Then
            If CLng(arrItemSource(lCounter)) > 0 Then
                lItemStatus = osItemToPipeline
            Else
                lItemStatus = osItemReleasedtoManf
            End If
        End If
    End If

```

```

                                Border.cls
End If

'update order item - call db function
lRow = oDOrderItem.UpdateOrderItemStatus(
                                alOrderID, lItemNumber,
CStr(lItemStatus))

'check for errors
If lRow <> 1 Then
    RaiseResError
DawnOrder.DawnOrderErrors.ecUpdateOrderItemStatusFailed, _
                                TypeName(Me), "IntUpdateOrderItemStatus"
End If

'if order item is shipped, then add shipping info, checked for pipelined
orders
If alOrderItemStatusCode = DawnOrder.DawnItemStatus.osItemShipped Or _
alOrderItemStatusCode =
DawnOrder.DawnItemStatus.osItemPipelineDropShipped Then
    Call UpdateOrderItemShippingInfo(alOrderID, lItemNumber, dShipdate,
sTrackNum)
    'If the item is in a pipeline, update pipeline order to Pipeline
Complete
    'if the item was drop shipped then to pipeline drop shipped
    If oROrder!drop_ship_ind = "1" Then
        'drop shipped
        lOrderItemStatusCode = osItemPipelineDropShipped
    Else
        'complete
        lOrderItemStatusCode = osItemPipelineComplete
    End If
    Set oRItems = oDOrderItem.GetOrderItem(alOrderID, lItemNumber)
    If Not IsNull(oRItems!pl_item_num) And (oRItems!pl_item_num >
0) Then
        bPipeline = True
        oRSup.AddNew
        oRSup!ord_id = oRItems!pl_ord_id
        oRSup!item_num = oRItems!pl_item_num
        oRSup!ship_dte = dShipdate
        oRSup!trk_num = sTrackNum
        oRSup!item_stus = lOrderItemStatusCode
        oRSup.Update
        Set oRManf = oManf.GetManfByID(oROrder!ord_src_mfr_id)
        sEmailNotes = sEmailNotes & vbCrLf &
GetPipelineEmailInfo(alOrderID, lItemNumber, True)
        sEmailNotes = sEmailNotes & vbCrLf &
GetItemsForEmail(alOrderID, dTotal, Array(lItemNumber)) & vbCrLf
        End If
    End If

    'enter a row in item history table
    AddOrderItemHistory alOrderID, lItemNumber, "Order Item Status was
Updated", asEmail, asIPAddress, lItemStatus

    'if item was pipeline cancelled or pipeline rejected, all orders items
pipelined from this item have to be stopped.
    If alProcess = 1 Then
        If alOrderItemStatusCode =
DawnOrder.DawnItemStatus.osItemPipelineCancelled Or _
alOrderItemStatusCode =
DawnOrder.DawnItemStatus.osItemPipelineRejected Then
            'find the order/item to which this item has been pipelined
to

```

```

Border.cls
        SSQl = " SELECT oi.ord_id , oi.item_num, o.mfr_id,
o.last_maint_dtm, o.ord_stus_cd, o.ord_stop_stus_cd,
        SSQl = SSQl & " oi.item_stus_cd, oi.item_stop_stus_cd "
        SSQl = SSQl & " FROM ord_hdr o, ord_item oi "
        SSQl = SSQl & " WHERE o.ord_id = oi.ord_id "
        'order can be on hold or active, item should be active
        SSQl = SSQl & " AND (o.ord_stop_stus_cd IS NULL OR
o.ord_stop_stus_cd = '6' ) AND oi.item_stop_stus_cd IS NULL "
        SSQl = SSQl & " AND oi.pl_ord_id = " & CStr(alOrderID) & "
AND oi.pl_item_num = " & CStr(lItemNumber)
        Set ORSPProcess = GetSQLRecordset(SSQl)
        If ORSPProcess.RecordCount > 0 Then
            bPipeline = True
            ORSDn.AddNew
            ORSDn!ord_id = ORSPProcess!ord_id
            ORSDn!item_num = ORSPProcess!item_num
            ORSDn.Update
            Set ORsManf = oManf.GetManfByID(ORSPProcess!mfr_id)
            sEmailNotes = sEmailNotes & vbCrLf &
GetPipelineEmailInfo(alOrderID, lItemNumber, False)
            sEmailNotes = sEmailNotes & vbCrLf &
GetItemsForEmail(alOrderID, dTotal, Array(lItemNumber)) & vbCrLf
        End If
    End If
End If
Next
'*****

'cancel order if all items in the order is cancelled or pipelineCancelled
If alOrderItemStatusCode = DawnOrder.DawnItemStatus.osItemCancelled Then
    CancelOrder alOrderID, oDBOrder, oDBOrderItem
End If

'if item is being moved from pipeline cancelled to some other item status, and
if order is cancelled, then release the order
If alOrderItemStatusCode <> DawnOrder.DawnItemStatus.osItemPipelineCancelled And
-
    lOrderStatus = DawnOrder.DawnOrderStatus.osCancelled Then
        Call ReleaseOrder(alOrderID, asInternalNotes, adOrderMaintDate, asEmail,
asIPAddress, asEmailNotes, 0, _
        "", "", alUserID)
    End If

'rule - if item status is shipped, check all other item statuses.
'if all items are shipped/dropshipped, mark the order status as Shipped else
'mark the order status as PartiallyShipped

If alOrderItemStatusCode = DawnOrder.DawnItemStatus.osItemShipped Or _
alOrderItemStatusCode = DawnOrder.DawnItemStatus.osItemPipelineDropShipped
Then
    'get order and items again
    Set ORSOrder = oDBOrder.GetOrder(alOrderID)
    Set ORSItems = oDBOrderItem.GetOrderItems(alOrderID)
    bItemNotShipped = False

    'check all the order item statuses for shipped
    If Not ORSItems.EOF Then
        Do Until ORSItems.EOF
            If (ORSItems("item_stus_cd") <>
DawnOrder.DawnItemStatus.osItemShipped) And _
                (ORSItems("item_stus_cd") <>
Page 23

```

```

                                BOrder.cls
DawnOrder.DawnItemStatus.osItemPipelineDropShipped) Then
    bItemNotShipped = True
    Exit Do
End If
ORSItems.MoveNext
Loop
End If

'boolean will be -ve if all items are cancelled, then set the
orderstatus to cancelled
If bItemNotShipped Then
    lNewOrderStatus = DawnOrder.DawnOrderStatus.osPartiallyShipped
Else
    lNewOrderStatus = DawnOrder.DawnOrderStatus.osShipped
End If

'update order status to shipped/partiallyShipped
lRow = oDBOrder.UpdateOrderStatus(alOrderID, CStr(lNewOrderStatus), _
                                ORSOrder("last_maint_dtm"))
'check error
If lRow <> 1 Then
    RaiseResError DawnOrder.DawnOrderErrors.ecUpdateOrderStatusFailed, _
                    TypeName(Me), "IntUpdateOrderItemStatus"
End If
End If

'call Db function to add order history
lHistoryID = AddOrderHistory(alOrderID, alEmailInd, aluserID, "", asEmail, 0,
asIPAddress, lOrderStatus)

'uday 8/15/00 - update all items in an order in one shot
'update pipeline orders (OrsUp)
If ORSUp.RecordCount > 0 Then
    ORSUp.Sort = "ord_id, item_num"
    ORSUp.MoveFirst
    loldID = ORSUp!ord_id
    lubound = -1
    Do while Not ORSUp.EOF
        If loldID <> ORSUp!ord_id And lubound >= 0 Then
            lubound = -1
            Set ORSProcess = oDBOrder.GetOrder(loldID)
            IntUpdateOrderItemStatus loldID, ArrItems, Array(), arrDate,
arrTrk, lOrderItemStatusCode, _
                                ""
ORSPProcess!last_maint_dtm, "", asIPAddress, "", 1, "", "", -1, '0
            loldID = ORSUp!ord_id
        End If
        lubound = lubound + 1
        ReDim Preserve ArrItems(lubound)
        ReDim Preserve arrDate(lubound)
        ReDim Preserve arrTrk(lubound)
        ArrItems(lubound) = ORSUp!item_num
        arrDate(lubound) = ORSUp!ship_dte
        arrTrk(lubound) = ORSUp!trk_num
        lOrderItemStatusCode = ORSUp!item_stus

        ORSUp.MoveNext
    Loop
    If lubound >= 0 Then
        Set ORSProcess = oDBOrder.GetOrder(loldID)
        IntUpdateOrderItemStatus loldID, ArrItems, Array(), arrDate, arrTrk,
lOrderItemStatusCode, _
                                "", ORSPProcess!last_maint_dtm,

```

```

Border.cls

"", asIPAddress, "", 1, "", "", -1, 0
End If

End If

'uday 8/15/00 - update all items in an order in one shot
'update pipeline orders (Orsdown)
If oRSDn.RecordCount > 0 Then
    oRSDn.Sort = "ord_id, item_num"
    oRSDn.MoveFirst
    loldID = oRSDn!ord_id
    lUbound = -1
    ReDim arrayitems(0)
    Do While Not oRSDn.EOF
        If loldID <> oRSDn!ord_id And lUbound >= 0 Then
            lUbound = -1
            Set ORSPProcess = oDBOrder.GetOrder(loldID)
            Call StopOrderItem(loldID, ArrItems, 5, "",
oRSPProcess!last_maint_dtm, "", asIPAddress, "", 1, "", "", -1, 1)
            loldID = oRSDn!ord_id
        End If
        lUbound = lUbound + 1
        ReDim Preserve ArrItems(lUbound)
        ArrItems(lUbound) = oRSDn!item_num
        oRSDn.MoveNext
    Loop
    If lUbound >= 0 Then
        Set ORSPProcess = oDBOrder.GetOrder(loldID)
        Call StopOrderItem(loldID, ArrItems, 5, "",
oRSPProcess!last_maint_dtm, "", asIPAddress, "", 1, "", "", -1, 1)
    End If
End If

'update notes
If abFromPipeline Then
    UpdateOrderNotes alOrderID, lHistoryID, asInternalNotes, asPipelineNotes, ""
Else
    UpdateOrderNotes alOrderID, lHistoryID, asInternalNotes, asPipelineNotes,
sEmailOriginal
End If

'send Email - based on pipeline
If bPipeline Then

    asEmail = IIf(IsNull(oRsManf!ord_email_addr),
oRsManf!admin_email_addr, oRsManf!ord_email_addr)
    asEmailNotes = sEmailNotes & vbCrLf & asEmailNotes

    'set the message for the Email
    Select Case alOrderItemStatusCode
    Case Is = osItemShipped, osItemPipelineDropShipped,
osItemPipelineComplete
        asEmailNotes = asEmailNotes & vbCrLf & "System Update: These
items were shipped."
    Case Is = osItemPipelineRejected, osItemPipelineCancelled,
osItemCancelled
        asEmailNotes = asEmailNotes & vbCrLf & "System Update: These
Items were cancelled."
    End Select

    'this is for mfr to mfr for a pipeline order
    sBody = CreateEmail(alOrderID, asEmail, emUpdateOrderItemStatus,
asEmailNotes, _

```

```

Border.cls
    alOrderItemStatusCode, , ,
bPipeline)
Else
    'not a pipeline order, this email is for regular mfr to customer
type
    If alOrderItemStatusCode = osItemShipped Or alOrderItemStatusCode =
osItemPipelineDropShipped Then
        If alProcess = 0 Then
            'this is thru pipeline process, get customer email ID
            Set oROrder = oDBOrder.GetOrder(alOrderID)
            Set oRSPProcess = GetSQLRecordset("SELECT email_addr FROM usr
WHERE usr_id = " & CStr(oROrder!usr_id))
            asEmail = oRSPProcess!email_addr
        End If
        sBody = CreateEmail(alOrderID, asEmail,
emUpdateOrderItemStatus, asEmailNotes, _
                                alOrderItemStatusCode,
arrItemNumber, , bPipeline)
    Else
        If (alEmailInd = 1) And (Len(Trim$(asEmail))) > 0) Then
            sBody = CreateEmail(alOrderID, asEmail,
emUpdateOrderItemStatus, asEmailNotes, _
                                alOrderItemStatusCode, arrItemNumber, , bPipeline)
        End If
    End If
End If

'get the order details
Set IntUpdateOrderItemStatus = oDBOrder.GetOrder(alOrderID)

'exit function
CtxSetComplete
Set oDBOrderItem = Nothing
Set oDBItemHist = Nothing
Set oDBOrder = Nothing
Set oDBordHist = Nothing
Set oManf = Nothing
Set oRManf = Nothing
Set oROrder = Nothing
Set oRItems = Nothing
Set oRSPProcess = Nothing
Set oRSDn = Nothing
Set oRSUP = Nothing

Exit Function

'error
ErrorHandler:
CtxSetAbort
Set oDBOrderItem = Nothing
Set oDBItemHist = Nothing
Set oDBOrder = Nothing
Set oDBordHist = Nothing
Set oManf = Nothing
Set oRManf = Nothing
Set oROrder = Nothing
Set oRItems = Nothing
Set oRSPProcess = Nothing
Set oRSDn = Nothing
Set oRSUP = Nothing

```

```

                                Border.cls
RaiseError TypeName(Me), "IntUpdateOrderItemStatus"
Exit Function
Resume
End Function

'=====
'this function stops an order status - cancel, reject, onhold
'input - order id, status, last maint date, email, ip address, email notes, email
indicator
' nothing
'=====
Public Sub StopOrder( _
    ByVal alOrderID As Long, ByVal alOrderStopCode As Long, ByVal asInternalNotes As
String,
    ByVal adMaintDate As Date, ByVal asEmail As String, ByVal asIPAddress As String,
    ByVal asEmailNotes As String, ByVal alEmailIndicator As Long, _
    ByVal asPipelineNotes As String, ByVal asShippingInstructions As String, _
    ByVal aluserID As Long)

    On Error GoTo ErrHandler

    Call IntStopOrder(alOrderID, alOrderStopCode, asInternalNotes, _
        adMaintDate, asEmail, asIPAddress, asEmailNotes, alEmailIndicator,
asPipelineNotes, asShippingInstructions, aluserID)

    SendEmails

    CtxSetComplete

    Exit Sub

'error
ErrHandler:
    CtxSetAbort
    RaiseError TypeName(Me), "StopOrder"
End Sub

Private Sub IntStopOrder( _
    ByVal alOrderID As Long, ByVal alOrderStopCode As Long, ByVal asInternalNotes As
String,
    ByVal adMaintDate As Date, ByVal asEmail As String, ByVal asIPAddress As String,
    ByVal asEmailNotes As String, ByVal alEmailInd As Long, ByVal asPipelineNotes As
String,
    ByVal asShippingInstructions As String, ByVal aluserID As Long)

    On Error GoTo ErrHandler

    Dim oDBord As DawnOrder.DBordHdr
    Dim oDBItem As DawnOrder.DBordItem
    Dim oDBhist As DawnOrder.DBordHist
    Dim lRow As Long
    Dim bPipeline As Boolean
    Dim oRS As ADODB.Recordset
    Dim dOrderMaintDate As Date
    Dim lorderID As Long
    Dim vItems() As Variant
    Dim v() As Variant
    Dim lItemCode As DawnOrder.DawnItemStatus
    Dim oRSPipe As ADODB.Recordset
    Dim ORSProcess As ADODB.Recordset

```

Border.cls

```

Dim sSQL As String
Dim lCounter As Long
Dim vPLItems() As Long
Dim lPLOrder, lPLCounter As Long
Dim lHistoryID As Long

'check for order status
If alOrderStopCode <> DawnOrder.DawnOrderStatus.osCancelled And _
    alOrderStopCode <> DawnOrder.DawnOrderStatus.osRejected And _
    alOrderStopCode <> DawnOrder.DawnOrderStatus.osOnhold Then
    RaiseResError DawnOrder.DawnOrderErrors.ecOrderStatusNotValid, TypeName(Me),
"IntStopOrder"
End If

'stop order
Set oDBOrd = CtxCreateObject("DawnOrder.DbOrdHdr")
Set oDBItem = CtxCreateObject("DawnOrder.DBOrdItem")
Set oDBhist = CtxCreateObject("DawnOrder.DbOrdHist")

lRow = oDBOrd.StopOrder(alOrderID, alOrderStopCode, adMaintDate)
If lRow <> 1 Then
    RaiseResError DawnOrder.DawnOrderErrors.ecUpdateOrderStatusFailed,
TypeName(Me), "IntStopOrder"
End If

'add history
lHistoryID = AddOrderHistory(alOrderID, alEmailInd, aluserID, "", asEmail, 0,
asIPAddress, alOrderStopCode)

'checked if the order is pipelined. If it is and the order is being rejected
'then reject the original items in the ref_num order.
'6/21/00, uday process changed, order can have multiple items for pipeline
'get the items, and now get pl_ord_id and pl_item_num from this items list and
update each pipelined item seperately
'if the current order was cancelled or rejected
'If this is the case do not send the email ( bPipeline = true )

Set oRS = oDBOrd.GetOrder(alOrderID)
If (Not IsNull(oRS!ord_src_mfr_id)) And oRS!ord_src_mfr_id > 0 Then
    bPipeline = True
    If (alOrderStopCode = osRejected) Or (alOrderStopCode = osCancelled)
Then
        If alOrderStopCode = osRejected Then
            lItemCode = osItemPipelineRejected
        Else
            lItemCode = osItemPipelineCancelled
        End If
        Set oRS = oDBItem.GetOrderItems(alOrderID)
        If Not oRS.EOF Then oRS.MoveFirst
        lCounter = -1
        lPLCounter = -1
        lPLOrder = -1
        lOrderID = -1
        Do While Not oRS.EOF
            If Not IsNull(oRS!pl_ord_id) And Not
                IsNull(oRS!pl_item_num) Then
                    If lOrderID <> oRS!pl_ord_id And lCounter >=
                        IntUpdateOrderItemStatus lOrderID,
vItems, v(), v(), v(), lItemCode, "", dOrderMaintDate, _
asEmail, asIPAddress, "", 0, "", "",
aluserID, 0

```


Border.cls

```

'send email
If alEmailInd And Not bPipeline Then
    CreateEmail alOrderID, asEmail, emStopOrder, asEmailNotes
End If

'complete
CtxSetComplete
Set oDBOrd = Nothing
Set oDBItem = Nothing
' Set oDBhist = Nothing
Set oRS = Nothing
Set oRSPipe = Nothing
Set oRSProcess = Nothing

Exit Sub

ErrorHandler:
CtxSetAbort

Set oDBOrd = Nothing
Set oDBItem = Nothing
' Set oDBhist = Nothing
Set oRS = Nothing
Set oRSPipe = Nothing
Set oRSProcess = Nothing

RaiseError TypeName(Me), "IntStopOrder"
End Sub

'=====
'this function releases an order status - cancel, reject, onhold
'input - order id, status, last maint date, email, ip address, email notes, email
indicator
' nothing
' uday 6/20/00 if order was pipelined, release is not allowed if order is not on hold
'=====
Public Sub ReleaseOrder( _
    ByVal alOrderID As Long, ByVal asInternalNotes As String, ByVal adMaintDate As
Date, _
    ByVal asEmail As String, ByVal asIPAddress As String, ByVal asEmailNotes As
String, _
    ByVal alEmailInd As Long, ByVal asPipelineNotes As String, _
    ByVal asShippingInstructions As String, ByVal aluserID As Long)

    On Error GoTo ErrorHandler

    Dim oDBOrd As DawnOrder.DBOrdHdr
    ' Dim oDBhist As DawnOrder.DBOrdHist
    Dim lRow As Long
    Dim oRS As ADODB.Recordset
    Dim lHistoryID As Long

    'release order
    Set oDBOrd = CtxCreateObject("DawnOrder.DbOrdHdr")
    ' Set oDBhist = CtxCreateObject("DawnOrder.DbOrdHist")

    'check for pipeline orders
    Set oRS = oDBOrd.GetOrder(alOrderID)
    If (Not IsNull(oRS!ord_src_mfr_id)) And (oRS!ord_stus_cd <> "6") Then
        RaiseResError ecPipelineOrderReleaseNotAllowed, TypeName(Me),
"ReleaseOrderItem"
    End If

```

Border.cls

```

    lRow = oDBord.ReleaseOrder(alOrderID, adMaintDate)
    If lRow <> 1 Then
        RaiseResError DawnOrder.DawnOrderErrors.ecUpdateOrderStatusFailed,
        TypeName(Me), "ReleaseOrder"
    End If

    Set oRS = oDBord.GetOrder(alOrderID)
    'add history
    lHistoryID = AddOrderHistory(alOrderID, alEmailInd, aluserID, "", asEmail, 0,
    asIPAddress, oRS!ord_stus_cd)

    'if email ind, then send email
    If alEmailInd = 1 Then
        CreateEmail alOrderID, asEmail, emUpdateOrderStatus, asEmailNotes
    End If

    'update notes
    UpdateOrderNotes alOrderID, lHistoryID, asInternalNotes, asPipelineNotes,
    asEmailNotes

    'complete
    CtxSetComplete
    Set oDBord = Nothing
    ' Set oDBhist = Nothing
    Set oRS = Nothing
    Exit Sub

ErrHandler:
    CtxSetAbort
    Set oDBord = Nothing
    ' Set oDBhist = Nothing
    Set oRS = Nothing
    RaiseError TypeName(Me), "ReleaseOrder"
End Sub

'=====
'this function stops an order item - cancel
'input - order id, item array, status, last maint date, email, ip address, email
'notes, email indicator
' nothing
'=====
Public Sub StopOrderItem( _
    ByVal alOrderID As Long, ByVal arrItemNumber As Variant, _
    ByVal alOrderItemStopCode As Long, ByVal asInternalNotes As String, _
    ByVal adOrderMaintDate As Date, ByVal asEmail As String, ByVal asIPAddress As
String, _
    ByVal asEmailNotes As String, ByVal alEmailInd As Long, _
    ByVal asPipelineNotes As String, ByVal asShippingInstructions As String, ByVal
aluserID As Long, _
    Optional ByVal alProcess As Long = 0)

    On Error GoTo ErrHandler

    Dim oDBord As DawnOrder.DBordHdr
    ' Dim oDBhist As DawnOrder.DBordHist
    Dim oDBitem As DawnOrder.DBordItem
    ' Dim oDBitemHist As DawnOrder.DBordItemHist
    Dim oRSorder As ADODB.Recordset
    Dim oRSItems As ADODB.Recordset
    Dim lRow As Long
    Dim lOrderStatus As Long
    Dim lCounter As Long

```

Border.cls

```

Dim bItemNotCancelled As Boolean
Dim oRSItem As ADODB.Recordset
Dim bPipeline, bPipelineDn As Boolean
Dim lManf As Long
Dim lHistoryID As Long

Dim lOrderID As Long
Dim vItem(0) As Variant
Dim dOrderMaintDate As Date
Dim oRSPipe As ADODB.Recordset

Dim sSQL As String
Dim oRSProcess As ADODB.Recordset
Dim sEmail, sEmailTxt, sMessage As String
Dim sEmailDn, sEmailtxtDn, sMessageDn As String
Dim oManf As DawnManf.UManf
Dim oRSManf As ADODB.Recordset
Dim dTotal As Double
Dim bOrderCancelled As Boolean

'record set for handling items in sequence, so that orders are updated together
instead of one item at a time
Dim oRSUp As ADODB.Recordset
Dim oRSDn As ADODB.Recordset
Dim arrItems() As Long
Dim lOldID As Long
Dim lUBound As Long

'create DB objects
Set oDBOrd = CtxCreateObject("DawnOrder.DbOrdHdr")
Set oDBHist = CtxCreateObject("DawnOrder.DbOrdHist")
Set oDBItem = CtxCreateObject("DawnOrder.DBOrdItem")
Set oDBItemHist = CtxCreateObject("DawnOrder.DBOrdItemHist")
Set oRSOrder = oDBOrd.GetOrder(alOrderID)
Set oManf = CtxCreateObject("DawnManf.UManf")
lOrderStatus = oRSOrder("ord_stus_cd")

'check for order status
If lOrderStatus = DawnOrder.DawnOrderStatus.osCancelled Or _
    lOrderStatus = DawnOrder.DawnOrderStatus.osRejected Or _
    lOrderStatus = DawnOrder.DawnOrderStatus.osOnhold Or _
    lOrderStatus = DawnOrder.DawnOrderStatus.osShipped Then
    RaiseResError DawnOrder.DawnOrderErrors.ecItemStatusChangeNotAllowed, _
        TypeName(Me), "StopOrderItem"
End If

'check for pipeline
lManf = oRSOrder!mfr_id
bPipeline = False
bPipelineDn = False

'set up pipeline record sets
Set oRSUp = GetClientRS()
oRSUp.Fields.Append "ord_id", adInteger
oRSUp.Fields.Append "item_num", adInteger
oRSUp.Open
Set oRSDn = GetClientRS()
oRSDn.Fields.Append "ord_id", adInteger
oRSDn.Fields.Append "item_num", adInteger
oRSDn.Open

'stop order items in the array - loop thru the array
For lCounter = 0 To UBound(arrItemNumber)

```

Border.cls

```

Set oRsItem = oDBItem.GetOrderItem(alOrderID, arrItemNumber(lCounter))

'update pipeline order item to pipeline cancelled, if this was called
directly i.e. alProcess=0
If alProcess = 0 Then
    If (Not IsNull(oRsItem!pl_ord_id)) And (Not
IsNull(oRsItem!pl_item_num)) Then
        If oRsItem!pl_ord_id > 0 And oRsItem!pl_item_num > 0 Then
            bPipeline = True
            lOrderID = oRsItem!pl_ord_id
            vItem(0) = oRsItem!pl_item_num
            Set oRsPipe = oDBOrd.GetOrder(lOrderID)
            dOrderMaintDate = oRsPipe!last_maint_dtm
            Set oRsManf =
oManf.GetManfByID(oRsOrder!ord_src_mfr_id)
            sEmail = IIf(IsNull(oRsManf!ord_email_addr),
oRsManf!admin_email_addr, oRsManf!ord_email_addr)
            sEmailTxt = sEmailTxt & vbCrLf &
GetPipelineEmailInfo(alOrderID, arrItemNumber(lCounter), True)
            sEmailTxt = sEmailTxt & vbCrLf &
GetItemsForEmail(alOrderID, dTotal, Array(arrItemNumber(lCounter))) & vbCrLf
            sMessage = "System Update : These items which
were in pipeline was cancelled by the source manufacturer." & _
" They are now marked as
Pipeline cancelled."

            oRSUP.AddNew
            oRSUP!ord_id = lOrderID
            oRSUP!item_num = oRsItem!pl_item_num
            oRSUP.Update

        End If
    End If
End If

'stop item
lRow = oDBItem.StopOrderItem(alOrderID, arrItemNumber(lCounter),
alOrderItemStopCode)
If lRow <> 1 Then
    RaiseResError
DawnOrder.DawnOrderErrors.ecUpdateOrderItemStatusFailed, _
TypeName(Me),
"StopOrderItem"
End If

'add history
AddOrderItemHistory alOrderID, arrItemNumber(lCounter), "System Update
Item Cancelled", asEmail, asIPAddress, alOrderItemStopCode

'if this item was pipelined to another order, go down the chain and stop
the item for all pipelined orders down the chain
'find the order/item to which this item has been pipelined to
sSQL = " SELECT oi.ord_id , oi.item_num, o.mfr_id, o.last_maint_dtm,
o.ord_stus_cd, o.ord_stop_stus_cd, "
sSQL = sSQL & " oi.item_stus_cd, oi.item_stop_stus_cd "
sSQL = sSQL & " FROM ord_hdr o, ord_item oi "
sSQL = sSQL & " WHERE o.ord_id = oi.ord_id "
'order can be on hold or active, item should be active
sSQL = sSQL & " AND (o.ord_stop_stus_cd IS NULL OR o.ord_stop_stus_cd =
'6' ) AND oi.item_stop_stus_cd IS NULL "
sSQL = sSQL & " AND oi.pl_ord_id = " & CStr(alOrderID) & " AND
oi.pl_item_num = " & CStr(arrItemNumber(lCounter))
Set oRSProcess = GetSQLRecordset(sSQL)
If oRSProcess.RecordCount > 0 Then

```

```

                                BOrder.cls
        bPipelineDn = True
        Set oRsManf = oManf.GetManfByID(oRsProcess!mfr_id)
        sEmailDn = IIf(IsNull(oRsManf!ord_email_addr),
oRsManf!admin_email_addr, oRsManf!ord_email_addr)
        sEmailtxtDn = sEmailtxtDn & vbCrLf & GetPipelineEmailInfo(alOrderID,
arrItemNumber(lCounter), False)
        sEmailtxtDn = sEmailtxtDn & vbCrLf & GetItemsForEmail(alOrderID,
dTTotal, Array(arrItemNumber(lCounter))) & vbCrLf
        sMessageDn = "System Update : These items were cancelled."
        oRSDn.AddNew
        oRSDn!ord_id = oRsProcess!ord_id
        oRSDn!item_num = oRsProcess!item_num
        oRSDn.Update
    End If

Next

'rule - after order item is updated
'check all the order items. if all items are cancelled, then the order must be
cancelled
borderCancelled = CancelOrder(alOrderID, oDBord, oDBItem)

'add history and notes
lHistoryID = AddOrderHistory(alOrderID, alEmailInd, aluserID, "", sEmail,
alEmailInd, asIPAddress, lOrderStatus)
UpdateOrderNotes alOrderID, lHistoryID, asInternalNotes, asPipelineNotes,
asEmailNotes

'uday 8/15/00 - update all items in an order in one shot
'update pipeline orders (OrsUp)
If oRSUp.RecordCount > 0 Then
    oRSUp.Sort = "ord_id, item_num"
    oRSUp.MoveFirst
    loldID = oRSUp!ord_id
    lUbound = -1
    Do While Not oRSUp.EOF
        If loldID <> oRSUp!ord_id And lUbound >= 0 Then
            lUbound = -1
            Set oRsProcess = oDBord.GetOrder(loldID)
            IntUpdateOrderItemStatus lOrderID, ArrItems, Array(), Array(),
Array(), osItemPipelineCancelled, sMessage, _
                oRsProcess!last_maint_dtm, "", asIPAddress, "", 1, "", "",
-1
                loldID = oRSUp!ord_id
        End If
        lUbound = lUbound + 1
        ReDim Preserve ArrItems(lUbound)
        ArrItems(lUbound) = oRSUp!item_num
        oRSUp.MoveNext
    Loop
    If lUbound >= 0 Then
        Set oRsProcess = oDBord.GetOrder(loldID)
        IntUpdateOrderItemStatus lOrderID, ArrItems, Array(), Array(),
Array(), osItemPipelineCancelled, sMessage, _
            oRsProcess!last_maint_dtm, "", asIPAddress, "", 1, "", "",
-1
    End If
End If

'uday 8/15/00 - update all items in an order in one shot
'update pipeline orders (OrsDn)
If oRSDn.RecordCount > 0 Then

```

```

Border.cls

oRSDn.Sort = "ord_id, item_num"
oRSDn.MoveFirst
lOldID = oRSDn!ord_id
lUbound = -1
ReDim arrayitems(0)
Do While Not oRSDn.EOF
    If lOldID <> oRSDn!ord_id And lUbound >= 0 Then
        lUbound = -1
        Set oRSProcess = oDBOrd.GetOrder(lOldID)
        Call StopOrderItem(lOldID, ArrItems, 5, sMessageDn,
oRSProcess!last_maint_dtm, "", asIPAddress, "", 1, "", "", -1, 1)
        lOldID = oRSDn!ord_id
    End If
    lUbound = lUbound + 1
    ReDim Preserve ArrItems(lUbound)
    ArrItems(lUbound) = oRSDn!item_num
    oRSDn.MoveNext
Loop
If lUbound >= 0 Then
    Set oRSProcess = oDBOrd.GetOrder(lOldID)
    Call StopOrderItem(lOldID, ArrItems, 5, sMessageDn,
oRSProcess!last_maint_dtm, "", asIPAddress, "", 1, "", "", -1, 1)
End If
End If

'send email if pipeline stop
If bPipeline Then
    asEmailNotes = sEmailTxt & vbCrLf & asEmailNotes & vbCrLf & sMessage
    Call CreateEmail(alOrderID, sEmail, emUpdateOrderItemStatus,
asEmailNotes, osItemCancelled, , , bPipeline)
ElseIf bPipelineDn Then
    asEmailNotes = sEmailtxtDn & vbCrLf & asEmailNotes & vbCrLf & sMessageDn
    Call CreateEmail(alOrderID, sEmailDn, emUpdateOrderItemStatus,
asEmailNotes, osItemCancelled, , , bPipeline)
Else
    'send email for regular orders if requested
    If (Not bOrderCancelled) And (alEmailInd = 1) And (Len(Trim$(asEmail))) > 0)
Then
        Call CreateEmail(alOrderID, asEmail,
emUpdateOrderItemStatus, asEmailNotes, _
osItemCancelled, arrItemNumber, , bPipeline)
    End If
End If

'complete
CtxSetComplete
Set oDBOrd = Nothing
Set oDBhist = Nothing
Set oDBItem = Nothing
Set oDBItemHist = Nothing
Set oRSorder = Nothing
Set oRSItems = Nothing
Set oRSPipe = Nothing
Set oRSProcess = Nothing
Set oRSDn = Nothing
Set oRSUP = Nothing

Exit Sub

ErrorHandler:
CtxSetAbort
Set oDBOrd = Nothing

```

Border.cls

```

' Set oDBhist = Nothing
' Set oDBItem = Nothing
' Set oDBItemHist = Nothing
' Set oRSorder = Nothing
' Set oRSItems = Nothing
' Set oRSpipe = Nothing
' Set oRSProcess = Nothing
' Set oRSDn = Nothing
' Set oRSUP = Nothing

RaiseError TypeName(Me), "StopOrderItem"
Exit Sub

End Sub

'=====
'this function releases an order item - cancel
'input - order id, item array, last maint date, email, ip address, email notes,
'email indicator
' nothing
'uday 5/30/00 if order is cancelled, allow item status change and then release the
order also
'uday 6/20/00 if order was pipelined, release is not allowed
'=====
Public Sub ReleaseOrderItem( _
ByVal alOrderID As Long, ByVal arrItemNumber As Variant, ByVal asInternalNotes
As String, _
ByVal adOrderMaintDate As Date, ByVal asEmail As String, ByVal asIPAddress As
String, _
ByVal asEmailNotes As String, ByVal alEmailInd As Long, ByVal asPipelineNotes As
String, _
ByVal asShippingInstructions As String, ByVal aluserID As Long)

On Error GoTo ErrHandler

Dim oDBOrd As DawnOrder.DBOrdHdr
' Dim oDBhist As DawnOrder.DBOrdHist
Dim oDBItem As DawnOrder.DBOrdItem
' Dim oDBItemHist As DawnOrder.DBOrdItemHist
Dim oRS As ADODB.Recordset
Dim lRow As Long
Dim lOrderStatus As Long
Dim lCounter As Long
Dim lHistoryID As Long

'create DB objects
Set oDBOrd = CtxCreateObject("DawnOrder.DbOrdHdr")
' Set oDBhist = CtxCreateObject("DawnOrder.DbOrdHist")
Set oDBItem = CtxCreateObject("DawnOrder.DBOrdItem")
' Set oDBItemHist = CtxCreateObject("DawnOrder.DBOrdItemHist")
Set oRS = oDBOrd.GetOrder(alOrderID)
lOrderStatus = oRS("ord_stus_cd")

If Not IsNull(oRS!ord_src_mfr_id) Then
RaiseResError ecPipelineOrderReleaseNotAllowed, TypeName(Me),
"ReleaseOrderItem"
End If

'check for order status
If lOrderStatus = DawnOrder.DawnOrderStatus.osRejected Or _
lOrderStatus = DawnOrder.DawnOrderStatus.osOnhold Or _
lOrderStatus = DawnOrder.DawnOrderStatus.osShipped Then
RaiseResError DawnOrder.DawnOrderErrors.ecItemStatusChangeNotAllowed, _

```



```

                                Border.cls
                                TypeName(Me), "ReleaseOrderItem"
End If

'release order items in the array - loop thru the array
For lCounter = 0 To UBound(arrItemNumber)
    'stop item
    lRow = oDBItem.ReleaseOrderItem(alOrderID, arrItemNumber(lCounter))
    If lRow <> 1 Then
        RaiseResError
DawnOrder.DawnOrderErrors.ecUpdateOrderItemStatusFailed, _
                                TypeName(Me),
"ReleaseOrderItem"
    End If

    Set oRS = GetSQLRecordset("SELECT item_stus_cd FROM ord_item WHERE
ord_id = " & _
                                alOrderID & " AND item_num = " & arrItemNumber(lCounter))

    'add history
    AddOrderItemHistory alOrderID, arrItemNumber(lCounter), "System Update
Item Released", _
                                asEmail, asIPAddress, oRS!item_stus_cd
Next

'uday 5/30/00 - if order status is cancelled, then release the order also
If lOrderStatus = DawnOrder.DawnOrderStatus.osCancelled Then
    Call ReleaseOrder(alOrderID, asInternalNotes, adOrderMaintDate, asEmail,
asIPAddress, asEmailNotes, alEmailInd, _
                                asPipelineNotes, asShippingInstructions, aluserID)
Else
    'add history and update
    Set oRS = GetSQLRecordset("SELECT ord_stus_cd FROM ord_hdr WHERE ord_id = "
& alOrderID)
    lHistoryID = AddOrderHistory(alOrderID, alEmailInd, aluserID, "", asEmail,
0, asIPAddress, oRS!ord_stus_cd)
    UpdateOrderNotes alOrderID, lHistoryID, asInternalNotes, asPipelineNotes,
asEmailNotes
    If alEmailInd = 1 Then
        CreateEmail alOrderID, asEmail, emUpdateOrderItemStatus, asEmailNotes, ,
arrItemNumber
    End If
End If

'complete
CtxSetComplete
Set oDBOrd = Nothing
' Set oDBHist = Nothing
' Set oDBItem = Nothing
' Set oDBItemHist = Nothing
Set oRS = Nothing
Exit Sub

ErrorHandler:
CtxSetAbort
Set oDBOrd = Nothing
' Set oDBHist = Nothing
' Set oDBItem = Nothing
' Set oDBItemHist = Nothing
Set oRS = Nothing
RaiseError TypeName(Me), "ReleaseOrderItem"
End Sub

```

Border.cls

```
'=====
' This is an obsolete function.  It was left for compatibility issues.
'=====
'Public Sub CheckAddress(ByVal alUserID As Long, ByVal alOrgID As Long, ByVal
alManfID As Long, _
'    ByVal asAddr1 As String, ByVal asAddr2 As String, _
'    ByVal asCity As String, ByVal asState As String, ByVal asZip As String, _
'    ByVal asCountry As String, ByVal Phone As String, ByVal Ext As String, _
'    ByVal AltPhone As String, ByVal AltExt As String)
'
'    Dim lEntityID As Long
'    Dim sEntityTyp_cd As String
'End Sub

'=====
'this function updates internal notes and pipeline notes
'input - columns for notes
'long - the number of rows affected by the SQL
'8/15/00 - this function now adds notes every time to the DB
'=====
Public Sub UpdateOrderNotes(ByVal alOrderID As Long, ByVal alOrderHistoryID As Long,
ByVal asInternalNotes As String, _
ByVal asPipelineNotes As String, ByVal
asEmailBody As String, _
Optional ByVal alSpecialInstructions As
String = "")

    On Error GoTo ErrHandler

    Dim oDB As DawnOrder.DBOrdNotes
    Dim oSeq As DawnSeq.BSequence
    Dim lRow As Long
    Dim lNotesID As Long

    Set oSeq = CtxCreateObject("DawnSeq.BSequence")
    Set oDB = CtxCreateObject("DawnOrder.DBOrdNotes")

    'add internal notes
    If Len(Trim$(asInternalNotes)) > 0 Then
        lNotesID = oSeq.GetNextInSequence("ord_notes")
        lRow = oDB.AddOrderNotes(lNotesID, alOrderID, alOrderHistoryID,
asInternalNotes, osInternal)
        If lRow <> 1 Then
            RaiseResError DawnOrder.DawnOrderErrors.ecUpdateOrderFailed,
TypeName(Me), "UpdateOrderNotes"
        End If
    End If

    'add pipeline notes
    If Len(Trim$(asPipelineNotes)) > 0 Then
        lNotesID = oSeq.GetNextInSequence("ord_notes")
        lRow = oDB.AddOrderNotes(lNotesID, alOrderID, alOrderHistoryID,
asPipelineNotes, osPipeline)
        If lRow <> 1 Then
            RaiseResError DawnOrder.DawnOrderErrors.ecUpdateOrderFailed,
TypeName(Me), "UpdateOrderNotes"
        End If
    End If

    'add email body
    If Len(Trim$(asEmailBody)) > 0 Then
        lNotesID = oSeq.GetNextInSequence("ord_notes")

```

```

Border.cls
    lRow = oDB.AddOrderNotes(lNotesID, alOrderID, alOrderHistoryID, asEmailBody,
osEmail)
    If lRow <> 1 Then
        RaiseResError DawnOrder.DawnOrderErrors.ecUpdateOrderFailed,
TypeName(Me), "UpdateOrderNotes"
    End If
End If

    'add special notes
    If Len(Trim$(alSpecialInstructions)) > 0 Then
        lNotesID = oSeq.GetNextInSequence("ord_notes")
        lRow = oDB.AddOrderNotes(lNotesID, alOrderID, alOrderHistoryID,
alSpecialInstructions, osSpecial)
        If lRow <> 1 Then
            RaiseResError DawnOrder.DawnOrderErrors.ecUpdateOrderFailed,
TypeName(Me), "UpdateOrderNotes"
        End If
    End If

    CtxSetComplete
    Set oSeq = Nothing
    Set oDB = Nothing
    Exit Sub

ErrorHandler:
    CtxSetAbort
    Set oSeq = Nothing
    Set oDB = Nothing
    RaiseError TypeName(Me), "UpdateOrderNotes"
    Exit Sub

End Sub

'=====
'this function updates the order item shipping info and date
'input - Order ID, item number , tracking number, ship date
'returns nothing
'=====
Public Sub UpdateOrderItemShippingInfo(ByVal alOrderID As Long, ByVal alItemNumber
As Long, _
ByVal adShipDate
As Date, ByVal asTrackNumber As String)
    On Error GoTo ErrorHandler

    'vars
    Dim oDB As DawnOrder.DBOrdItem
    Dim lRow As Long

    'check date
    If Trim$(asTrackNumber) > "" Then
        If adShipDate <= 0 Then
            RaiseResError DawnOrder.DawnOrderErrors.ecInvalidShipdate, TypeName(Me),
"UpdateOrderItemShippingInfo"
        End If
    End If

    'call DB function
    Set oDB = CtxCreateObject("DawnOrder.DBOrdItem")
    lRow = oDB.UpdateOrderItemShippingInfo(alOrderID, alItemNumber, adShipDate,
asTrackNumber)

    'check error
    If lRow <> 1 Then

```

```

Border.cls
RaiseResError DawnOrder.DawnOrderErrors.ecUpdateOrderItemFailed,
TypeName(Me), "UpdateOrderItemShippingInfo"
End If

'run sql
CtxSetComplete
Set oDB = Nothing
Exit Sub

'error
ErrorHandler:
CtxSetAbort
Set oDB = Nothing
RaiseError TypeName(Me), "UpdateOrderItemShippingInfo"
End Sub

```

```

'=====
'This function is a wrapper of the StopOrderItem function. The Only difference
'is that this function finds out the last_maint_dtm.
'uday 6/21/00, this is obsolete, this function is taken care of thru update
orderitem
'=====
'Public Sub StopPipelineOrderItem(ByVal alOrderID As Long, ByVal alItemNumber As
Long, _
'    ByVal alOrderItemStopCode As Long, ByVal asEmail As String, ByVal asIPAddress
As String, _
'    ByVal asPipelineNotes As String, ByVal aluserID As Long)
'
'    Err.Raise 1, , "Obsolete function."
'    RaiseError TypeName(Me), "Do not use this function."
'
'    'TODO ErrorHandling and clean up
'    Dim vItem(0) As Long
'    Dim vItems As Variant
'    Dim oDBOrderHeader As DawnOrder.DBOrdHdr
'    Dim oRS As ADODB.Recordset
'    Dim dMaintDate As Date
'
'    If alOrderItemStopCode = osItemCancelled Then alOrderItemStopCode =
osItemPipelineCancelled
'    vItem(0) = alItemNumber
'    vItems = vItem()
'
'    'call db comp function
'    Set oDBOrderHeader = CtxCreateObject("DawnOrder.DBOrdHdr")
'    Set oRS = oDBOrderHeader.GetOrder(alOrderID)
'    dMaintDate = oRS!last_maint_dtm
'
'    StopOrderItem alOrderID, vItems, alOrderItemStopCode, Null, dMaintDate,
asEmail, asIPAddress, Null, 0, asPipelineNotes, Null, aluserID
'End Sub

```

```

'=====
'This function creates all e-mails and adds them to the e-mail queue. It returns
'the e-mail text to be added to the notes.
'=====
Private Function CreateEmail(ByVal alOrderID As Long, ByVal asToEmail As String, _
    ByVal alType As emEmail, Optional ByVal asEmailNotes As String, _
    Optional alOrderItemStatusCode As Long, Optional arrItemNumber As
Variant, _

```

```

                                Border.cls
Optional aOrgID As Long, Optional ByVal abPipe As Boolean = False) As
String
    Dim sBody As String
    Dim oRsManf As ADODB.Recordset
    Dim oRsOrder As ADODB.Recordset
    Dim oRsOrg As ADODB.Recordset
    Dim dTotal As Double
    Dim sURL As String
    Dim oManf As DawnManf.UManf
    Dim lManfID As Long
    Dim sSubject As String
    Dim lOrderID As Long
    Dim arrEmail(5) As String
    Dim sFromEmail As String
    Dim oDBOrderHeader As DawnOrder.DBOrdHdr
    Dim sManfName As String
    Dim bSendToManf As Boolean
    Dim sSQL As String
    Dim oRsNotes As ADODB.Recordset
    Dim oDBNotes As DawnOrder.DBOrdNotes
    Dim sShipping As String

    On Error GoTo ErrorHandler

    arrEmail(4) = vbNullString
    arrEmail(5) = vbNullString

    'call db comp function
    Set oDBOrderHeader = CtxCreateObject("DawnOrder.DBOrdHdr")
    Set oRsOrder = oDBOrderHeader.GetOrder(aOrderID)
    Set oDBNotes = CtxCreateObject("DawnOrder.DBOrdNotes")
    Set oRsNotes = oDBNotes.GetOrderNotes(aOrderID, Array(1))
    If oRsNotes.RecordCount >= 1 Then
        sShipping = Trim$(oRsNotes!notes)
    End If
    sURL = oRsOrder!shop_url
    Set oDBOrderHeader = Nothing
    lManfID = oRsOrder!mfr_id
    Set oManf = CtxCreateObject("DawnManf.UManf")
    Set oRsManf = oManf.GetManfByID(lManfID)
    lOrderID = aOrderID

    'get admin Email if to Email is empty ie send Email to Mfr
    If Len(Trim$(asToEmail)) = 0 Then
        If alType = emAddOrder Then
            bSendToManf = True
            If Not IsNull(oRsManf("ord_email_addr")) Then
                asToEmail = oRsManf("ord_email_addr")
            Else
                If Not IsNull(oRsManf("admin_email_addr")) Then asToEmail =
oRsManf("admin_email_addr")
                End If
            Else
                If Not IsNull(oRsManf("admin_email_addr")) Then asToEmail =
oRsManf("admin_email_addr")
                End If
            End If
        End If

        'get order Email for mfr
        If Not IsNull(oRsManf("ord_email_addr")) Then
            sFromEmail = oRsManf("ord_email_addr")
        Else

```

```

                                Border.cls
sFromEmail = oRsManf("admin_email_addr")
End If

Select Case alType
    Case Is = emAddOrder
        If bSendToManf Then 'email to mfr
            sBody = "Date/Time of Order: " & oRsOrder!ord_dte & "" & vbCrLf &
vbCrLf
            sBody = sBody & "Order Number: " & oRsOrder!ord_num & vbCrLf &
vbCrLf
            sBody = sBody & GetItemsForEmail(lOrderID, dTotal)
            sBody = sBody & "Tax : " & Format(oRsOrder!calc_tax, "Currency") &
vbCrLf
            sBody = sBody & "Shipping : " & Format(oRsOrder!calc_ship_cost,
"Currency") & vbCrLf & vbCrLf
            dTotal = dTotal + CDb1(oRsOrder!calc_tax) +
Cdb1(oRsOrder!calc_ship_cost)
            sBody = sBody & "Order Total : " & Format(CStr(dTotal), "Currency")
& vbCrLf & vbCrLf
            sBody = sBody & "-----" &
vbCrLf
            sBody = sBody & "Special Instructions: " & vbCrLf
            If Trim(sShipping) = "" Then
                sBody = sBody & "<None>" & vbCrLf & vbCrLf
            Else
                sBody = sBody & sShipping & vbCrLf & vbCrLf
            End If
            sBody = sBody & "-----" &
vbCrLf & vbCrLf & "OnlineCustom... Bringing e-Business to your business" & vbCrLf
            sBody = sBody & "===== " &
vbCrLf

            sSubject = "OLC new order notification " & oRsOrder!ord_num
            'Finds the Org id to Cc
            If alOrgID > 0 Then
                On Error Resume Next
                sSQL = "SELECT email_addr FROM org WHERE org_id = " & alOrgID
                Set oRsOrg = GetSQLRecordset(sSQL)
                If Err = 0 Then
                    If Not oRsOrg.EOF And (Not IsNull(oRsOrg!email_addr)) Then
                        arrEmail(4) = oRsOrg!email_addr
                    End If
                End If
                Err.Clear
                Set oRsOrg = Nothing
                On Error GoTo ErrorHandler
            End If
        Else 'email to user
            lOrderID = oRsOrder!ord_id

            sURL = oRsOrder!shop_url

            'set up body for Email
            sBody = "Thank you for your order! This e-mail confirms that your
order has " & _
                "been received by " & oRsManf!mfr_nme & " on " &
oRsOrder!ord_dte & "." & vbCrLf & vbCrLf
            sBody = sBody & "Your order confirmation number is: " &
oRsOrder!ord_num & vbCrLf & vbCrLf
            sBody = sBody & "The items in your order are:" & vbCrLf & vbCrLf

            sBody = sBody & GetItemsForEmail(lOrderID, dTotal)

            sBody = sBody & "Tax : " & Format(oRsOrder!calc_tax, "Currency") &

```

Border.cls

```

vbCrLf
    sBody = sBody & "Shipping : " & Format(ORsOrder!calc_ship_cost,
"Currency") & vbCrLf & vbCrLf
    dTotal = dTotal + Cdbl(ORsOrder!calc_tax) +
Cdbl(ORsOrder!calc_ship_cost)
    sBody = sBody & "Order Total : " & Format(CStr(dTotal), "Currency")
& vbCrLf & vbCrLf
    sBody = sBody & "-----" &
vbCrLf
    sBody = sBody & "Special Instructions: " & vbCrLf
    If Trim(sShipping) = "" Then
        sBody = sBody & "<None>" & vbCrLf & vbCrLf
    Else
        sBody = sBody & sShipping & vbCrLf & vbCrLf
    End If
    sBody = sBody & "-----" &
vbCrLf & vbCrLf
    sBody = sBody & "Pending credit verification, your order will be
processed and " & _
    "shipped shortly. If for any reason we cannot process your
order as submitted" & _
    ", you will receive an e-mail notification explaining the
item(s) in question." & vbCrLf & vbCrLf
    sBody = sBody & "When your order is shipped we will notify you by
e-mail. If you " & _
    "have any concerns or questions please feel free to contact
us at:" & vbCrLf & vbCrLf

    sBody = sBody & ORsManf!mfr_nme & vbCrLf
    sBody = sBody & ORsManf!addr_1 & vbCrLf
    If Not IsNull(ORsManf!addr_2) Then
        If Len(Trim(ORsManf!addr_2)) > 0 Then
            sBody = sBody & ORsManf!addr_2 & vbCrLf
        End If
    End If
    sBody = sBody & ORsManf!city & ", " & ORsManf!st_prov & " " &
ORsManf!zip_pstl_cd & vbCrLf
    sBody = sBody & ORsManf!Phone & vbCrLf
    sBody = sBody & ORsManf!fax & vbCrLf
    sBody = sBody & ORsManf!admin_email_addr & vbCrLf & vbCrLf

    If Not abPipe Then
        sBody = sBody & "You can track the status of your order at our
website, " & CheckURL(SURL) & ", " & _
        "by logging in to your member account and clicking on CHECK
ORDER STATUS " & _
        "at the top of the page." & vbCrLf & vbCrLf
    End If
    sBody = sBody & "Thanks for choosing " & ORsManf!mfr_nme & "." &
vbCrLf
    sBody = sBody & "===== " &
vbCrLf
    sSubject = ORsManf!mfr_nme & " Order Confirmation: " &
ORsOrder!ord_num
    End If
    Case Is = emStopOrder
        sBody = "Your order confirmation number is: " & ORsOrder!ord_num &
vbCrLf & vbCrLf
        sBody = sBody & asEmailNotes & vbCrLf & vbCrLf
        sBody = sBody & ". If you have any concerns or questions please feel
free to contact us at:" & vbCrLf & vbCrLf

```

'If the order comes from a kiosk then use kiosk info.

```

Border.cls
If ORSOrder!on_behalf_of_mfr_id > 0 Then
    sBody = sBody & GetManfInfo(ORSOrder!on_behalf_of_mfr_id, sManfName,
SURL)
Else
    sBody = sBody & GetManfInfo(ORSOrder!mfr_id, sManfName, SURL)
End If
sSubject = sManfName & " note about your order " & ORSOrder!ord_num
Case IS = emUpdateOrderStatus
    sBody = "Your order confirmation number is: " & ORSOrder!ord_num &
vbCrLf & vbCrLf
    sBody = sBody & asEmailNotes & vbCrLf & vbCrLf
    'If the order comes from a kiosk then use kiosk info.
    If ORSOrder!on_behalf_of_mfr_id > 0 Then
        sBody = sBody & GetManfInfo(ORSOrder!on_behalf_of_mfr_id, sManfName,
SURL)
    Else
        sBody = sBody & GetManfInfo(ORSOrder!mfr_id, sManfName, SURL)
    End If
    sSubject = sManfName & " note about your order " & ORSOrder!ord_num
Case IS = emUpdateOrderItemStatus
    If alOrderItemStatusCode = osItemShipped Then
        sBody = "This is to inform you that the following items have been
shipped for order: "
    Else
        sBody = "Your order number is: "
    End If
    sBody = sBody & ORSOrder!ord_num & vbCrLf & vbCrLf
    If Len(asEmailNotes) > 0 Then sBody = sBody & asEmailNotes & vbCrLf &
vbCrLf

    'add items to email body if not pipeline email
    If Not abPipe Then
        sBody = sBody & GetItemsForEmail(lOrderID, dTotal, arrItemNumber)
    End If

    If alOrderItemStatusCode = osItemShipped Then
        If ORSOrder!ord_stus_cd = osShipped Then
            sBody = sBody & "This shipment completes your order." & vbCrLf &
vbCrLf
        ElseIf ORSOrder!ord_stus_cd = osPartiallyShipped Then
            sBody = sBody & "This is a partial shipment." & vbCrLf & vbCrLf
        End If
    End If

    sBody = sBody & "If you have any concerns or questions please feel free
to contact us at:" & vbCrLf & vbCrLf
    sBody = sBody & GetManfInfo(lManfID, sManfName, SURL, abPipe)
    If alOrderItemStatusCode = osItemShipped Then
        sBody = sBody & vbCrLf & vbCrLf & "Thanks for choosing " & sManfName
& "."
        sSubject = sManfName & " shipment notice for order " &
ORSOrder!ord_num
    Else
        sSubject = sManfName & " note about your order " & ORSOrder!ord_num
    End If

End Select

CreateEmail = sBody

```


Border.cls

```

arrEmail(0) = sFromEmail
arrEmail(1) = asToEmail
arrEmail(2) = sSubject
arrEmail(3) = sBody

If m_colEmail Is Nothing Then Set m_colEmail = New Collection
m_colEmail.Add arrEmail()

Set oDBOrderHeader = Nothing
Set oManf = Nothing
Set oRSManf = Nothing
Set oRSOrder = Nothing
Set oRSOrg = Nothing
Set oRSNotes = Nothing
Set oDBNotes = Nothing

Exit Function

ErrorHandler:
End Function

'=====
' This function adds the items section to an e-mail
'=====
Private Function GetItemsForEmail(ByVal aOrderID As Long, dTotal As Double,
Optional arrItemNumber As Variant) As String

Dim oRSItem As ADODB.Recordset
Dim oUOrder As DawnOrder.UOrder
Dim dPrice As Double
Dim sParts As String
Dim bSkip As Boolean

Set oUOrder = CtxCreateObject("DawnOrder.UOrder")
Set oRSItem = oUOrder.GetAllOrders(, , , , , , , , , aOrderID)
oRSItem.MoveFirst
Do Until oRSItem.EOF
    If Not IsMissing(arrItemNumber) Then
        If Not CheckPart(oRSItem("item_num"), arrItemNumber) Then bSkip = True
    End If
    If Not bSkip Then
        dPrice = Cdbl(oRSItem("ord_qty")) * Cdbl(oRSItem("unit_price"))
        dTotal = dTotal + dPrice
        sParts = sParts & "Part: " & oRSItem("part_num") & vbCrLf & _
            "Description: " & oRSItem("brief_descr") & vbCrLf & _
            "Qty: " & oRSItem("ord_qty") & vbCrLf & _
            "Price: $" & CStr(dPrice) & vbCrLf & vbCrLf
    End If
    bSkip = False
    oRSItem.MoveNext
Loop
Set oUOrder = Nothing
GetItemsForEmail = sParts
End Function

'=====
' This function finds out if an item should be included in the e-mail.
'=====
Private Function CheckPart(sPart As String, arrItemNumber As Variant) As Boolean

Dim lCounter As Long

```

```

Border.cls
For lCounter = 0 To UBound(arrItemNumber)
    If arrItemNumber(lCounter) = sPart Then
        CheckPart = True
    End If
Next

End Function

'=====
' this function gets order items
' input - order ID
' record set with the order items
'=====
Public Function GetOrderItems(ByVal alOrderID As Long) As ADODB.Recordset
    On Error GoTo ErrHandler

    'var
    Dim oDBItem As DawnOrder.DBOrdItem
    Dim oProd As DawnProd.UProd
    Dim oRSItems As ADODB.Recordset
    Dim oRSPart As ADODB.Recordset
    Dim oRSCustom As ADODB.Recordset

    'check order ID
    1 If alOrderID <= 0 Then
    2     RaiseResError DawnOrder.DawnOrderErrors.ecOrderIDNotValid, TypeName(Me),
"GetOrderItems"
    3 End If

    'create objs
    4 Set oDBItem = CtxCreateObject("DawnOrder.DBOrdItem")
    5 Set oProd = CtxCreateObject("DawnProd.UProd")

    'call db comp function
    6 Set oRSItems = oDBItem.GetOrderItems(alOrderID)

    'add items to custom RS
    7 Set oRSCustom = GetClientRS()

    'add custom columns
    8 oRSCustom.Fields.Append "ord_id", adInteger
    9 oRSCustom.Fields.Append "item_num", adInteger
    10 oRSCustom.Fields.Append "part_id", adInteger
    11 oRSCustom.Fields.Append "pl_ord_id", adInteger
    12 oRSCustom.Fields.Append "pl_item_num", adInteger
    13 oRSCustom.Fields.Append "item_stus_cd", adInteger
    14 oRSCustom.Fields.Append "item_stop_stus_cd", adInteger
    15 oRSCustom.Fields.Append "ord_qty", adInteger
    16 oRSCustom.Fields.Append "ship_qty", adInteger
    17 oRSCustom.Fields.Append "ret_qty", adInteger
    18 oRSCustom.Fields.Append "unit_price", adCurrency
    19 oRSCustom.Fields.Append "design_id", adInteger
    20 oRSCustom.Fields.Append "create_dtm", adDate
    21 oRSCustom.Fields.Append "last_maint_dtm", adDate
    22 oRSCustom.Fields.Append "part_num", adVarChar, 50
    23 oRSCustom.Fields.Append "int_part_num", adVarChar, 50
    24 oRSCustom.Fields.Append "part_nme", adVarChar, 50
    25 oRSCustom.Fields.Append "commodity_num", adVarChar, 50
    26 oRSCustom.Fields.Append "brief_descr", adVarChar, 255
    27 oRSCustom.Fields.Append "descr", adVarChar, 255
    28 oRSCustom.Fields.Append "sugg_list_price", adCurrency
    29 oRSCustom.Fields.Append "img_url", adVarChar, 255
    30 oRSCustom.Fields.Append "ship_dte", adDate

```

```

Border.cls
'31 ORSCustom.Fields.Append "ship_track_num", adVarChar, 30
'32 ORSCustom.Fields.Append "rev_num", adInteger
'33 ORSCustom.Fields.Append "mfr_nme", adVarChar, 50
'34 ORSCustom.Fields.Append "src_mfr_id", adInteger
'
'open custom RS
'35 ORSCustom.Open
'
'36 If Not ORSItems.EOF Then
'
'37     ORSItems.MoveFirst
'38     Do Until ORSItems.EOF
'39         ORSCustom.AddNew
'
'         'add order item data
'40         ORSCustom("ord_id") = ORSItems("ord_id")
'41         ORSCustom("item_num") = ORSItems("item_num")
'42         ORSCustom("part_id") = ORSItems("part_id")
'43         If Not IsNull(ORSItems("src_mfr_id")) Then
'44             ORSCustom("src_mfr_id") = ORSItems("src_mfr_id")
'45         End If
'46         If Not IsNull(ORSItems("pl_ord_id")) Then
'47             ORSCustom("pl_ord_id") = ORSItems("pl_ord_id")
'48         End If
'49         If Not IsNull(ORSItems("pl_item_num")) Then
'50             ORSCustom("pl_item_num") = ORSItems("pl_item_num")
'51         End If
'52         If Not IsNull(ORSItems("src_mfr_id")) Then
'53             If ORSItems("src_mfr_id") = 0 Then
'54                 ORSCustom("mfr_nme") = "<internal manufacturing>"
'55             Else
'56                 ORSCustom("mfr_nme") = ORSItems("mfr_nme")
'57             End If
'58         End If
'59         ORSCustom("item_stus_cd") = ORSItems("item_stus_cd")
'60         ORSCustom("ord_qty") = ORSItems("ord_qty")
'61         ORSCustom("ship_qty") = ORSItems("ship_qty")
'62         ORSCustom("ret_qty") = ORSItems("ret_qty")
'63         ORSCustom("unit_price") = ORSItems("unit_price")
'64         If Not IsNull(ORSItems("design_id")) Then
'65             ORSCustom("design_id") = ORSItems("design_id")
'66         End If
'67         ORSCustom("create_dtm") = ORSItems("create_dtm")
'68         ORSCustom("last_maint_dtm") = ORSItems("last_maint_dtm")
'
'69         If Not IsNull(ORSItems("ship_track_num")) Then
'70             ORSCustom("ship_track_num") = ORSItems("ship_track_num")
'71         End If
'72         If Not IsNull(ORSItems("ship_dte")) Then
'73             ORSCustom("ship_dte") = ORSItems("ship_dte")
'74         End If
'75         If Not IsNull(ORSItems("rev_num")) Then
'76             ORSCustom("rev_num") = ORSItems("rev_num")
'77         End If
'78
'79         'get part data
'80         Set ORSPart = oProd.GetPart(ORSItems("part_id"))
'81
'82         'set part data
'83         If Not ORSPart.EOF Then
'84             ORSCustom("part_num") = ORSPart("part_num")
'85             ORSCustom("int_part_num") = ORSPart("int_part_num")
'86             ORSCustom("part_nme") = ORSPart("part_nme")

```

```

'87         Border.cls
'88         oRsCustom("commodity_num") = oRSPart("commodity_num")
'89         oRsCustom("brief_descr") = oRSPart("brief_descr")
'90         oRsCustom("descr") = oRSPart("descr")
'91         If Not IsNull(oRSPart("sugg_list_price")) Then
'92             oRsCustom("sugg_list_price") = oRSPart("sugg_list_price")
'93         End If
'94         If Not IsNull(oRSPart("img_url")) Then
'95             oRsCustom("img_url") = oRSPart("img_url")
'96         End If
'97     End If
'98     'move to next order item
'99     oRSItems.MoveNext
'100 Loop
'101 End If
'102
'103 'exit
'104 Set GetOrderItems = oRsCustom
'105 CtxSetComplete
'106 Set oDBItem = Nothing
'107 Set oProd = Nothing
'108 Set oRSItems = Nothing
'109 Set oRSPart = Nothing
'110 Set oRsCustom = Nothing
'111 Exit Function
'112
'113 'error
'114 ErrHandler:
'115 CtxSetAbort
'116 Set oDBItem = Nothing
'117 Set oProd = Nothing
'118 Set oRSItems = Nothing
'119 Set oRSPart = Nothing
'120 Set oRsCustom = Nothing
'121 RaiseError TypeName(Me), "GetOrderItems: " & Err
'122 End Function
'123
'124 =====
'125 This function adds the footer with the manf info to an e-mail.
'126 =====
Private Function GetManfInfo(lManfID As Long, Optional sManfName As String, _
Optional asURL As String, Optional abPipe As
Boolean = False) As String
    Dim oManf As DawnManf.UManf
    Dim oRsManf As ADODB.Recordset
    Dim oRsManfURL As ADODB.Recordset
    Dim sURL As String
    Dim sBody As String

    Set oManf = CtxCreateObject("DawnManf.UManf")
    Set oRsManf = oManf.GetManfByID(lManfID)

    sManfName = oRsManf!mfr_nme
    sBody = oRsManf!mfr_nme & vbCrLf
    sBody = sBody & oRsManf!addr_1 & vbCrLf
    If Not IsNull(oRsManf!addr_2) Then
        If Len(Trim$(oRsManf!addr_2)) > 0 Then
            sBody = sBody & oRsManf!addr_2 & vbCrLf
        End If
    End If
End If

```

```

        Border.cls
        sBody = sBody & oRsManf!city & ", " & oRsManf!st_prov & " " &
oRsManf!zip_pstl_cd & vbCrLf
        sBody = sBody & "Phone: " & oRsManf!Phone & vbCrLf
        sBody = sBody & "Fax: " & oRsManf!fax & vbCrLf
        sBody = sBody & oRsManf!admin_email_addr & vbCrLf & vbCrLf

        If Not abPipe Then
            sBody = sBody & "You can track the status of your order at our website, " &
CheckURL(asURL) & ", " & _
                "by logging in to your member account and clicking on CHECK ORDER STATUS
" & _
                "at the top of the page."
        End If

        GetManfInfo = sBody

        Set oManf = Nothing
        Set oRsManf = Nothing
        Set oRsManfUrl = Nothing

```

End Function

'Left Public for compatibility issues. Should be private.
Public Sub SendEmails()

```

    Dim v As Variant
    Dim oMail As DawnMail.Umail
    Dim i As Integer

    Set oMail = CtxCreateObject("DawnMail.Umail")

    If Not m_colEmail Is Nothing Then
        If m_colEmail.Count > 0 Then
            For Each v In m_colEmail
                Call oMail.SendMessage(v(0), v(1), v(2), v(3), v(4), v(5)) '
(sFromEmail, asToEmail, asSubject, sBody)
            Next

            For i = 1 To m_colEmail.Count
                m_colEmail.Remove (1)
            Next

        End If
    End If

    Set oMail = Nothing

    Exit Sub

```

End Sub

```

'=====
' This function makes sure that an URL begins with http://
'=====
Private Function CheckURL(sURL As String) As String

```

```

    If Left$(sURL, 7) = "http://" Then
        CheckURL = sURL
    Else
        CheckURL = "http://" & sURL
    End If

```

End Function

```

'=====
'this function adds src mfr to items who have only 1 source mfr
'this function is called by other functions within this level only , so no set
complete or abort is done here
'args - order ID
'no returns
'=====
Private Sub SetSourceMfr(ByVal alManfID As Long, ByVal alOrderID As Long)
On Error GoTo ErrHandler

    Dim oRSItems As ADODB.Recordset
    Dim oRSsrc As ADODB.Recordset
    Dim oDB As DawnOrder.DBOrdItem
    Dim oProd As DawnProd.UProd
    Dim lItemNum As Long
    Dim lPart As Long
    Dim lSrcMfr As Long
    Dim lRow As Long

    Set oProd = CtxCreateObject("DawnProd.UProd")
    Set oDB = CtxCreateObject("DawnOrder.DBOrdItem")
    Set oRSItems = oDB.GetOrderItems(alOrderID)

    If Not oRSItems.EOF Then
        oRSItems.MoveFirst
        Do While Not oRSItems.EOF
            lItemNum = oRSItems!item_num
            lPart = oRSItems!part_id
            Set oRSsrc = oProd.GetSources(alManfID, -1, -1, lPart)

            'set src only if part has only 1 src or if the dflt src ind = 1
            If oRSsrc.RecordCount = 1 Then
                lSrcMfr = oRSsrc!src_mfr_id
                lRow = oDB.UpdateOrderItemSource(alOrderID, lItemNum,
lSrcMfr)
                If lRow <> 1 Then
                    RaiseResError DawnOrder.ecUpdateOrderItemFailed,
TypeName(Me), "SetSourceMfr"
                End If
            Else
                If Not oRSsrc.EOF Then
                    Do While Not oRSsrc.EOF
                        If oRSsrc!dflt_src_ind = "1" Then
                            lSrcMfr = oRSsrc!src_mfr_id
                            lRow =
oDB.UpdateOrderItemSource(alOrderID, lItemNum, lSrcMfr)
                            If lRow <> 1 Then
                                RaiseResError
DawnOrder.ecUpdateOrderItemFailed, TypeName(Me), "SetSourceMfr"
                            End If
                        End If
                        oRSsrc.MoveNext
                    Loop
                End If
            End If
            oRSItems.MoveNext
        Loop
    End If

    Set oDB = Nothing
    Set oRSItems = Nothing

```

Border.cls

```

Set oRsSrc = Nothing
Set oProd = Nothing
Exit Sub

```

ErrorHandler:

```

RaiseError TypeName(Me), "SetSourceMfr"
Set oDB = Nothing
Set oRSItems = Nothing
Set oRsSrc = Nothing
Set oProd = Nothing
Exit Sub

```

End Sub

```

'=====
'this function gets the pipeline que and adds all the items in the que into pipeline
orders
'args - pipeline que ID
'no return
'=====

```

```

Public Sub Pipeline(ByVal alQueID As Long, ByVal aluserID As Long)
On Error GoTo ErrorHandler

```

```

    Dim oP As Object
    Dim oManf As DawnManf.UManf
    Dim oRsManf As ADODB.Recordset
    Dim oRsQue As ADODB.Recordset
    Dim oContext AsObjectContext
    Dim oRate As DawnRate.URate ' to calculate shipping and taxes
    Dim sDropShipInd As String
    Dim sShipName, sShipAddr1, sShipAddr2, sShipCity, sShipState, sShipZip,
sShipCountry As String
    Dim sShipPhone, sShipExt, sShipDescr, sShipBriefDescr, sShiptoCompanyName As
String
    Dim sBillName, sBillAddr1, sBillAddr2, sBillCity, sBillState, sBillZip,
sBillCountry As String
    Dim sBillPhone, sBillExt, sBillDescr, sBillBriefDescr, sBilltoCompanyName As
String
    Dim lOldOrderID As Long
    Dim bFirstRow As Boolean
    Dim lOrdSrcMfrID, lManfID As Long
    Dim sMfrEmail As String
    Dim sPO, sRef, sURL As String
    Dim sCCName, sCCNum, sCCDate As String
    Dim oRsPipeItems As ADODB.Recordset
    Dim lPipeOrdID, lPipeItemNum, lPartID, lQty, lDesignID As Long
    Dim cPrice As Currency
    Dim oRsOrder As ADODB.Recordset
    Dim oDB As DawnOrder.DBOrdHdr
    Dim oDBItem As DawnOrder.DBOrdItem
    Dim oRsNotes As ADODB.Recordset
    Dim lOrderID() As Long
    Dim dOrderLastMaintdate() As Date
    Dim lAddGroupMessage(), lAddDropShipMessage(), lAddDirectShipmessage() As Long
    Dim lCounter, lCtr As Long
    Dim oRsNewOrder As ADODB.Recordset
    Dim sNotes, sGroupMessage, sDropShipMessage As String
    Dim sMfrGroupMessage, sDirectShipMessage, sItemMessage() As String
    Dim cTax, cShip, cTotal As Currency
    Dim lShipID, lStateID As Long
    Dim oRsTax As ADODB.Recordset
    Dim dPlDate As Date
    Dim soldDropShip As String

```

Border.cls

```

Dim lItemArray() As Long
Dim lUbound As Long
Dim sInternalNotes, sEmailTxt As String
Dim bFirstDropShip As Boolean

'disable commit
Set oContext = GetObjectContext
If Not oContext Is Nothing Then
    oContext.DisableCommit
End If

'set up counters
lCounter = 0
lUbound = -1

'create DB level
Set oDB = CtxCreateObject("DawnOrder.DBOrdHdr")
Set oDBItem = CtxCreateObject("DawnOrder.DBOrdItem")

Set oP = CtxCreateObject("DawnPipeQueue.UPipeQueue")
Set oRSQue = oP.GetAllPLQueueInfo(alQueID)
Set oRSNotes = oP.GetNotes(alQueID)

'create rate for tax and shipping
Set oRate = CtxCreateObject("DawnRate.URate")

'check to make sure there is atleast one items to be pipelined
If (oRSQue Is Nothing) Or (oRSQue.RecordCount <= 0) Then
    RaiseResError ecNoItemsInPipe, TypeName(Me), "Pipeline"
Else
    oRSQue.MoveFirst
    dPlDate = oRSQue!last_maint_dtm
End If

'get mfr notes
If Not oRSNotes.EOF Then
    oRSNotes.MoveFirst
    Do While Not oRSNotes.EOF
        If oRSNotes!pl_note_typ_cd = "1" Then
            sMfrGroupMessage = oRSNotes!notes
        Else
            sDirectShipMessage = oRSNotes!notes
        End If
        oRSNotes.MoveNext
    Loop
End If

'create manf for manf info
Set oManf = CtxCreateObject("DawnManf.UManf")

'initialize first row
bFirstRow = True
bFirstDropShip = True
sGroupMessage = "GROUP MESSAGE : "
sDropShipMessage = "DROP SHIP : "

'set up items record set for pipling items to the new pipeline orders
Set oRSPipeItems = GetClientRS()
oRSPipeItems.Fields.Append "part_id", adInteger
oRSPipeItems.Fields.Append "pipe_ord_id", adInteger
oRSPipeItems.Fields.Append "pipe_item_num", adInteger
oRSPipeItems.Fields.Append "ord_qty", adInteger
oRSPipeItems.Fields.Append "unit_price", adCurrency

```



```

Border.cls
oRSPipeItems.Fields.Append "design_id", adInteger
oRSPipeItems.Open

'remove any rows from pipe items, so that it is a clean empty recordset
Do While oRSPipeItems.RecordCount > 0
    oRSPipeItems.MoveFirst
    oRSPipeItems.Delete adAffectCurrent
Loop

'sort the ors so that all drop ship items are listed in the end and get drop
ship indicator
oRSQue.Sort = "drop_ship_ind, ord_id, item_num"
oRSQue.MoveFirst
lOrdSrcMfrID = oRSQue!mfr_id
Set oRSManf = oManf.GetManfByID(lOrdSrcMfrID)

'set up the orders for pipeline, going thru each of the items in the pipe que
and getting relavant data
Do While Not oRSQue.EOF

    soldDropShip = sDropShipInd
    sDropShipInd = IIf(IsNull(oRSQue!drop_ship_ind), "0",
CStr(oRSQue!drop_ship_ind))
    If sDropShipInd = "0" Then
        If bFirstRow Then
            'add earlier order that may have been
            'the sort will make sure that dropship = 0
            'does
            'not happen, this will take care of the
            If (Not oRSPipeItems Is Nothing) And
                'set up tax and shipping
                'calculate the total for all items
                oRSPipeItems.MoveFirst
                Do While Not oRSPipeItems.EOF
                    cTotal = cTotal +
                    oRSPipeItems.MoveNext
                Loop
                Set oRSTax =
                cTax = oRSTax!tax_amt
                cShip = oRSTax!ship_amt
                oRSPipeItems.MoveFirst
                sURL = "No URL"
                Set oRSNewOrder =
                sShipCity, sShipState, sShipZip, sShipCountry, _
                sBillCity, sBillState, sBillZip, sBillCountry, _
                sCCNum, sCCDate, soldDropShip, 0,
                sShipBriefDescr, cTax, cShip, "", sMfrEmail, "",
                sShipPhone, "", sBillPhone,
                sBillExt, "", "", sBillBriefDescr, sBillDescr, sShipDescr, _
                sShipToCompanyName,
                sBillToCompanyName, 0, sCCName, sURL, oRSPipeItems)
                ReDim Preserve lOrderID(lCounter -
1)
                ReDim Preserve

```

```

Border.cls

dOrderLastMaintdate(lCounter - 1)
oRSNewOrder!ord_id
oRSNewOrder!last_maint_dtm
CStr(oRSNewOrder!ord_num) & " , "
sDropShipMessage = sDropShipMessage & CStr(oRSNewOrder!ord_num) & " , "

order - this should be for drop ship items.
separate order for the pipe mfr
the order
that it is a clean empty recordset
0

adAffectCurrent

oRSQue!po_num)
oRSQue!ref_num)
oRSQue!ship_typ_id)
oRSQue!url)
IIf(IsNull(oRSManf!ord_email_addr), oRSManf!admin_email_addr,
oRSManf!ord_email_addr)

lAddDirectShipmessage(lCounter)

get added

message per order

Border.cls

lOrderID(lCounter - 1) =
dOrderLastMaintdate(lCounter - 1) =
sGroupMessage = sGroupMessage &
If sOldDropShip = "1" Then
Set oRSNewOrder = Nothing
End If
'collect order data for the next set of
'here each order in the pipe que becomes one
'since the shipp address are unique only for
'remove any rows from pipe items, so
Do While oRSPipeItems.RecordCount >
oRSPipeItems.MoveFirst
oRSPipeItems.Delete

Loop
lManfID = oRSQue!src_mfr_id
sPO = IIf(IsNull(oRSQue!po_num), "",
sRef = IIf(IsNull(oRSQue!ref_num), "",
lShipID = IIf(IsNull(oRSQue!ship_typ_id), 0,
sURL = "" ' IIf(IsNull(oRSQue!url), "",
sMfrEmail =
'make sure notes need to be added
ReDim Preserve lAddGroupMessage(lCounter)
ReDim Preserve lAddDropShipMessage(lCounter)
ReDim Preserve
If oRSQue!incl_all_note_ind = "1" Then
lAddGroupMessage(lCounter) = 1
Else
lAddGroupMessage(lCounter) = 0
End If
'for direct ship, drop ship message does not
lAddDropShipMessage(lCounter) = 0
If sDropShipInd = "0" Then
lAddDirectShipmessage(lCounter) = 1
Else
lAddDirectShipmessage(lCounter) = 0
End If
ReDim Preserve sItemMessage(lCounter) ' one
lCounter = lCounter + 1

Else
lOldOrderID = oRSQue!ord_id
End If

```

```

Border.cls

Else
row, this ensures that all items in this order
bFirstDropShip Then
the same
(ORSPipeItems.RecordCount > 0) Then
in the order
ORSPipeItems!unit_price
ORRate.CalcTaxAndShipRatesByStProvAbbrev(lManfID, sShipState, lShipID, cTotal)
IntAddOrder(aluserID, lOrdSrcMfrID, -1, lManfID, -1, SPO, sRef, _
sShipCity, sShipState, sShipZip, sShipCountry, _
sBillCity, sBillState, sBillZip, sBillCountry, _
sShipBriefDescr, cTax, cShip, "", sMfrEmail, "", _
sBillExt, "", "", sBillBriefDescr, sBillDescr, sShipDescr, _
sBilltoCompanyName, 0, sCCName, sURL, ORSPipeItems)
1)
dOrderLastMaintdate(lCounter - 1)
ORSNewOrder!ord_id
ORSNewOrder!last_maint_dtm
CStr(ORSNewOrder!ord_num) & " , "
sDropShipMessage = sDropShipMessage & CStr(ORSNewOrder!ord_num) & " , "
that it is a clean empty recordset

'check if order ID is the same as the earlier
'goes into one order for pipe mfr for drop ship
If lOldOrderID = ORSQue!ord_id And Not
'add just the item because the order will be
Else
'add earlier order
If (Not ORSPipeItems Is Nothing) And
'set up tax and shipping
'calculate the total for all items
ORSPipeItems.MoveFirst
Do While Not ORSPipeItems.EOF
cTotal = cTotal +
ORSPipeItems.MoveNext
Loop
Set ORSTax =
cTax = ORSTax!tax_amt
cShip = ORSTax!ship_amt
ORSPipeItems.MoveFirst
sURL = "No URL"
Set ORSNewOrder =
sShipName, sShipAddr1, sShipAddr2,
sBillName, sBillAddr1, sBillAddr2,
sCCNum, sCCDate, sOldDropShip, 0,
sShipPhone, "", sBillPhone,
sShipDescr, _
sShipToCompanyName,
ReDim Preserve lOrderID(lCounter -
ReDim Preserve
lOrderID(lCounter - 1) =
dOrderLastMaintdate(lCounter - 1) =
sGroupMessage = sGroupMessage &
If sOldDropShip = "1" Then
Set ORSNewOrder = Nothing
End If
'collect order data for the next order
bFirstRow = True
bFirstDropShip = False
ReDim Preserve sItemMessage(lCounter)
sItemMessage(lCounter) = ""
'remove any rows from pipe items, so
Do While ORSPipeItems.RecordCount > 0
ORSPipeItems.MoveFirst
ORSPipeItems.Delete adAffectCurrent

```

```

Border.cls
    Loop
        lManfID = oRsQue!src_mfr_id
        SPO = IIf(IsNull(oRsQue!po_num), "",
oRsQue!po_num)
        sRef = IIf(IsNull(oRsQue!ref_num), "",
oRsQue!ref_num)
        lShipID = IIf(IsNull(oRsQue!ship_typ_id), 0,
oRsQue!ship_typ_id)
        sURL = "" ' IIf(IsNull(oRsQue!url), "",
oRsQue!url)
        sMfrEmail =
IIf(IsNull(oRsManf!ord_email_addr), oRsManf!admin_email_addr,
oRsManf!ord_email_addr)

        'make sure notes need to be added
        ReDim Preserve lAddGroupMessage(lCounter)
        ReDim Preserve lAddDropShipMessage(lCounter)
        ReDim Preserve
lAddDirectShipmessage(lCounter)
        If oRsQue!incl_all_note_ind = "1" Then
            lAddGroupMessage(lCounter) = 1
        Else
            lAddGroupMessage(lCounter) = 0
        End If
        If oRsQue!incl_drop_ship_note_ind = "1" Then
            lAddDropShipMessage(lCounter) = 1
        Else
            lAddDropShipMessage(lCounter) = 0
        End If
        If sDropShipInd = "0" Then
            lAddDirectShipmessage(lCounter) = 1
        Else
            lAddDirectShipmessage(lCounter) = 0
        End If
        lOldOrderID = oRsQue!ord_id
        ReDim Preserve sItemMessage(lCounter) ' one
message per order
        lCounter = lCounter + 1
    End If
End If

'for each order get ship and bill data, if drop ship, get
the ship address from the order table
'if direct ship, get the ship address from the pl que record
set
' all bill address comes from the pl que record set
If bFirstRow Then
    If sDropShipInd = "0" Then
        sShipName = IIf(IsNull(oRsQue!ship_to_nme), "",
CStr(oRsQue!ship_to_nme))
        sShipAddr1 = IIf(IsNull(oRsQue!ship_to_addr_1),
"", CStr(oRsQue!ship_to_addr_1))
        sShipAddr2 = IIf(IsNull(oRsQue!ship_to_addr_2),
"", CStr(oRsQue!ship_to_addr_2))
        sShipCity = IIf(IsNull(oRsQue!ship_to_city), "",
CStr(oRsQue!ship_to_city))
        sShipState = IIf(IsNull(oRsQue!ship_to_st_prov),
"", CStr(oRsQue!ship_to_st_prov))
        sShipZip =
IIf(IsNull(oRsQue!ship_to_zip_pstl_cd), "", CStr(oRsQue!ship_to_zip_pstl_cd))
        sShipCountry =
IIf(IsNull(oRsQue!ship_to_country), "", CStr(oRsQue!ship_to_country))
        sShipPhone = IIf(IsNull(oRsQue!ship_to_phone),

```

```

Border.cls
"", CStr(oRsQue!ship_to_phone))
CStr(oRsQue!ship_to_ext))
CStr(oRsQue!ship_descr))
IIf(IsNull(oRsQue!ship_brief_descr), "", CStr(oRsQue!ship_brief_descr))
IIf(IsNull(oRsQue!ship_to_company_nme), "", CStr(oRsQue!ship_to_company_nme))
Else
Set oRsOrder = oDB.GetOrder(oRsQue!ord_id)
sShipName = IIf(IsNull(oRsOrder!ship_to_nme),
"", CStr(oRsOrder!ship_to_nme))
sShipAddr1 =
IIf(IsNull(oRsOrder!ship_to_addr_1), "", CStr(oRsOrder!ship_to_addr_1))
sShipAddr2 =
IIf(IsNull(oRsOrder!ship_to_addr_2), "", CStr(oRsOrder!ship_to_addr_2))
sShipCity = IIf(IsNull(oRsOrder!ship_to_city),
"", CStr(oRsOrder!ship_to_city))
sShipState =
IIf(IsNull(oRsOrder!ship_to_st_prov), "", CStr(oRsOrder!ship_to_st_prov))
sShipZip =
IIf(IsNull(oRsOrder!ship_to_zip_pstl_cd), "", CStr(oRsOrder!ship_to_zip_pstl_cd))
sShipCountry =
IIf(IsNull(oRsOrder!ship_to_country), "", CStr(oRsOrder!ship_to_country))
sShipPhone = IIf(IsNull(oRsOrder!ship_to_phone),
"", CStr(oRsOrder!ship_to_phone))
sShipExt = IIf(IsNull(oRsOrder!ship_to_ext), "",
CStr(oRsOrder!ship_to_ext))
sShipDescr = IIf(IsNull(oRsOrder!ship_descr),
"", CStr(oRsOrder!ship_descr))
sShipBriefDescr =
IIf(IsNull(oRsOrder!ship_brief_descr), "", CStr(oRsOrder!ship_brief_descr))
sShiptoCompanyName =
IIf(IsNull(oRsOrder!ship_to_company_nme), "", CStr(oRsOrder!ship_to_company_nme))
Set oRsOrder = Nothing
End If
sBillName = IIf(IsNull(oRsQue!bill_to_nme), "",
CStr(oRsQue!bill_to_nme))
sBillAddr1 = IIf(IsNull(oRsQue!bill_to_addr_1), "",
CStr(oRsQue!bill_to_addr_1))
sBillAddr2 = IIf(IsNull(oRsQue!bill_to_addr_2), "",
CStr(oRsQue!bill_to_addr_2))
sBillCity = IIf(IsNull(oRsQue!bill_to_city), "",
CStr(oRsQue!bill_to_city))
sBillState = IIf(IsNull(oRsQue!bill_to_st_prov), "",
CStr(oRsQue!bill_to_st_prov))
sBillZip = IIf(IsNull(oRsQue!bill_to_zip_pstl_cd),
"", CStr(oRsQue!bill_to_zip_pstl_cd))
sBillCountry = IIf(IsNull(oRsQue!bill_to_country),
"", CStr(oRsQue!bill_to_country))
sBillPhone = IIf(IsNull(oRsQue!bill_to_phone), "",
CStr(oRsQue!bill_to_phone))
sBillExt = IIf(IsNull(oRsQue!bill_to_ext), "",
CStr(oRsQue!bill_to_ext))
sBillDescr = IIf(IsNull(oRsQue!bill_descr), "",
CStr(oRsQue!bill_descr))
sBillBriefDescr =
IIf(IsNull(oRsQue!bill_brief_descr), "", CStr(oRsQue!bill_brief_descr))
sBilltoCompanyName =
IIf(IsNull(oRsQue!bill_to_company_nme), "", CStr(oRsQue!bill_to_company_nme))

```

bFirstRow = False

```

                                Border.cls
                                End If

                                'add items to the pipe ORS all direct ship items go into one
order for the pipe mfr
                                ORSPipeItems.AddNew
                                ORSPipeItems!part_id = ORSQue!part_id
                                ORSPipeItems!pipe_ord_id = ORSQue!ord_id
                                ORSPipeItems!pipe_item_num = ORSQue!item_num
                                ORSPipeItems!ord_qty = ORSQue!ord_qty
                                ORSPipeItems!unit_price = ORSQue!unit_price
                                If Not IsNull(ORSQue!design_id) Then
                                    ORSPipeItems!design_id = ORSQue!design_id
                                End If
                                ORSPipeItems.Update

                                'set up item messages from pipeline que to add to new order
message
                                sItemMessage(lCounter - 1) = sItemMessage(lCounter - 1) & _
IIf(IsNull(ORSQue!pl_notes), "", ORSQue!pl_notes & vbCrLf)
                                ORSQue.MoveNext
                                Loop

                                'after the loop is complete, there will be one order left to be added to the
pipe mfr this will take care of that order
                                'add last order
                                If (Not ORSPipeItems Is Nothing) And (ORSPipeItems.RecordCount > 0) Then
                                    'set up tax and shipping
                                    'calculate the total for all items in the order
                                    ORSPipeItems.MoveFirst
                                    Do While Not ORSPipeItems.EOF
                                        cTotal = cTotal + ORSPipeItems!unit_price
                                        ORSPipeItems.MoveNext
                                    Loop
                                    Set ORSTax = ORRate.CalcTaxAndShipRatesByStProvAbbrev(lManfID,
sShipState, lShipID, cTotal)
                                    cTax = ORSTax!tax_amt
                                    cShip = ORSTax!ship_amt
                                    ORSPipeItems.MoveFirst
                                    sURL = "No URL"
                                    Set ORSNewOrder = IntAddOrder(aluserID, lOrdSrcMfrID, -1, lManfID,
-1, sPO, sRef, _
sShipName, sShipAddr1, sShipAddr2, sShipCity, sShipState, sShipZip,
sShipCountry, _
sBillName, sBillAddr1, sBillAddr2, sBillCity, sBillState, sBillZip,
sBillCountry, _
sCCNum, sCCDate, sDropShipInd, "0", sShipBriefDescr, cTax, cShip,
"", sMfrEmail, "", "", "",
sShipPhone, "", sBillPhone, sBillExt, "", "", sBillBriefDescr,
sBillDescr, sShipDescr, _
sShiptoCompanyName, sBilltoCompanyName, 0, sCCName, sURL,
ORSPipeItems)
                                    ReDim Preserve lOrderID(lCounter - 1)
                                    ReDim Preserve dOrderLastMaintdate(lCounter - 1)
                                    lOrderID(lCounter - 1) = ORSNewOrder!ord_id
                                    dOrderLastMaintdate(lCounter - 1) = ORSNewOrder!last_maint_dtm
                                    lCounter = lCounter + 1
                                    sGroupMessage = sGroupMessage & CStr(ORSNewOrder!ord_num) & " , "
                                    If sDropShipInd = "1" Then
                                        sDropShipMessage = sDropShipMessage & CStr(ORSNewOrder!ord_num)
& " , "
                                    End If
                                    Set ORSNewOrder = Nothing

```

```

End If

'setup notes for the orders
'if at least 1 order was created lcounter is same as UBound for lorderID ()
array
If UBound(lorderID) >= 0 Then
    sGroupMessage = Trim$(Left(sGroupMessage, Len(sGroupMessage) - 2))
    sDropShipMessage = Trim$(Left(sDropShipMessage, Len(sDropShipMessage) - 2))
    Do While lCtr <= (UBound(lorderID))
        'set up notes based on what the indicators
        sNotes = ""
        If lAddGroupMessage(lCtr) = 1 Then sNotes = sNotes & sGroupMessage &
vbCrLf
            sNotes = sNotes & smfrGroupMessage & vbCrLf
            If lAddDropShipMessage(lCtr) = 1 Then sNotes = sNotes & sDropShipMessage
& vbCrLf
                If lAddDirectShipmessage(lCtr) = 1 Then sNotes = sNotes &
sDirectShipMessage & vbCrLf
                    sNotes = sNotes & sItemMessage(lCtr)
                    'Call updateOrderNotes(lorderID(lCtr), "", "",
dOrderLastMaintdate(lCtr), sNotes, aluserID, 1)
                    lCtr = lCtr + 1
                Loop
            End If

'delete que items, error is raised in que, so no error handling here
Call op.DeletePipeQueueData(alqueID, dPlDate, False, aluserID)

'update all items for pipeline mfr to InPipeline
oRsQue.MoveFirst
oRsQue.Sort = ""
oRsQue.Sort = "ord_id, item_num"
oRsQue.MoveFirst
lOldOrderID = oRsQue!ord_id
lUbound = -1
Do While Not oRsQue.EOF

    'update old order
    If lOldOrderID <> oRsQue!ord_id Then

        Set oRsOrder = oDB.GetOrder(lOldOrderID)
        If lUbound >= 0 Then
            'no email needs to be sent here since, an Email will be
sent by src mfr to pipe mfr about the pipeline orders
            IntUpdateOrderItemStatus lOldOrderID, lItemArray,
Array(), _
                Array(), Array(), 7, sInternalNotes,
oRsOrder!last_maint_dtm, "", "", sEmailTxt, 0, "", "", aluserID
            End If
            lUbound = -1
            lOldOrderID = oRsQue!ord_id
        End If

        'add to item array
        lUbound = lUbound + 1
        ReDim Preserve lItemArray(lUbound)
        lItemArray(lUbound) = oRsQue!item_num
        sInternalNotes = oRsQue!int_notes
        sEmailTxt = IIf(IsNull(oRsQue!email_text), "", oRsQue!email_text)
        oRsQue.MoveNext
    End If
Loop

```

```

Border.cls
Set oRSorder = oDB.GetOrder(loldorderID)
If lUbound >= 0 Then
    'no email needs to be sent here since, an Email will be sent by src mfr
to pipe mfr about the pipeline orders
    IntUpdateOrderItemStatus loldorderID, lItemArray, Array(), _
    Array(), Array(), 7, sInternalNotes, oRSorder!last_maint_dtm, "", "",
sEmailTxt, 0, "", "", aluserID, , True
End If

'commit
If Not oContext Is Nothing Then
    oContext.EnableCommit
End If
CtxSetComplete

Set oP = Nothing
Set oRSQue = Nothing
Set oManf = Nothing
Set oRate = Nothing
Set oRSManf = Nothing
Set oRSPipeItems = Nothing
Set oContext = Nothing
Set oRSorder = Nothing
Set oDB = Nothing
Set oRSNotes = Nothing
Set oRSNewOrder = Nothing
Set oRSTax = Nothing
Set oDBItem = Nothing

Exit Sub

```

ErrorHandler:

```

'rollback
If Not oContext Is Nothing Then
    oContext.EnableCommit
End If
CtxSetAbort

RaiseError TypeName(Me), "Pipeline"
Set oP = Nothing
Set oRSQue = Nothing
Set oManf = Nothing
Set oRate = Nothing
Set oRSManf = Nothing
Set oRSPipeItems = Nothing
Set oContext = Nothing
Set oRSorder = Nothing
Set oDB = Nothing
Set oRSNotes = Nothing
Set oRSNewOrder = Nothing
Set oRSTax = Nothing
Set oDBItem = Nothing

Exit Sub

```

End Sub

```

'=====
'this function generates a message for order reference for pipeline order emails
'input - order ID, item number , pipline direction (is the email going up the chain
or doen the chain ?)

```


Border.cls

'this is for email messages only, the errors here should be ignored as they are not
part of the critical order flow process
'returns a string with reference message

Private Function GetPipelineEmailInfo(ByVal aOrdID As Long, ByVal aItemNum As
Long, ByVal abPipeDirUp As Boolean) As String
On Error Resume Next

Dim sRef As String
Dim sSQL As String
Dim oRS As ADODB.Recordset

sRef = "The order reference numbers for this pipelined order are as below:" &
vbCrLf

If abPipeDirUp Then

'get pipeline mfr details
sSQL = " SELECT o.ord_num, m.mfr_nme FROM ord_hdr o, mfr m "
sSQL = sSQL & " WHERE o.mfr_id = m.mfr_id "
sSQL = sSQL & " AND o.ord_id IN "
sSQL = sSQL & " (SELECT pl_ord_id FROM ord_item "
sSQL = sSQL & " WHERE ord_id = " & CStr(aOrdID) & " AND item_num = " &
CStr(aItemNum) & ") "
Set oRS = GetSQLRecordset(sSQL)
If oRS.RecordCount >= 1 Then
sRef = sRef & " For " & oRS!mfr_nme & ", the order rerefence is #" &
CStr(oRS!ord_num) & "." & vbCrLf
Set oRS = Nothing
End If

'get src mfr details
sSQL = " SELECT o.ord_num, m.mfr_nme FROM ord_hdr o, mfr m "
sSQL = sSQL & " WHERE o.mfr_id = m.mfr_id "
sSQL = sSQL & " AND o.ord_id = " & CStr(aOrdID)
Set oRS = GetSQLRecordset(sSQL)
If oRS.RecordCount = 1 Then
sRef = sRef & " For " & oRS!mfr_nme & ", the order rerefence is #" &
CStr(oRS!ord_num) & "." & vbCrLf
Set oRS = Nothing
End If

Else

'get the src mfr details
sSQL = " SELECT o.ord_num, m.mfr_nme FROM ord_hdr o, ord_item oi, mfr m "
sSQL = sSQL & " WHERE o.mfr_id = m.mfr_id AND o.ord_id = oi.ord_id "
sSQL = sSQL & " AND oi.pl_ord_id = " & CStr(aOrdID) & " AND oi.pl_item_num
= " & CStr(aItemNum)
sSQL = sSQL & " AND (o.ord_stop_stus_cd IS NULL OR o.ord_stop_stus_cd = '6'
) "
sSQL = sSQL & " AND oi.item_stop_stus_cd IS NULL "
Set oRS = GetSQLRecordset(sSQL)
If oRS.RecordCount = 1 Then
sRef = sRef & " For " & oRS!mfr_nme & ", the order rerefence is #" &
CStr(oRS!ord_num) & "." & vbCrLf
Set oRS = Nothing
End If

'get pipeline mfr details
sSQL = " SELECT o.ord_num, m.mfr_nme FROM ord_hdr o, mfr m "
sSQL = sSQL & " WHERE o.mfr_id = m.mfr_id "
sSQL = sSQL & " AND o.ord_id = " & CStr(aOrdID)

```

Border.cls

Set oRS = GetSQLRecordset(ssSQL)
If oRS.RecordCount = 1 Then
    sRef = sRef & " For " & oRS!mfr_nme & ", the order rerefence is #" &
CStr(oRS!ord_num) & "." & vbCrLf
    Set oRS = Nothing
End If

End If

Set oRS = Nothing
GetPipelineEmailInfo = sRef
Err.Clear
Exit Function

End Function

'=====
'this function gets a unque order history ID and adds a history row
'args - order ID, description, and all attributes for history
'returns the order history ID
'=====
Private Function AddOrderHistory(ByVal alOrderID As Long, ByVal alEmailInd As Long,
ByVal alUsrID As Long, _
ByVal asDescription As String, ByVal
asEmailAddr As String, _
ByVal alEmailTrkCode As Long, ByVal asIP As
String, _
ByVal alOrderStatus As Long) As Long
On Error GoTo ErrHandler

Dim oDB As DawnOrder.DBordHist
Dim oSeq As DawnSeq.BSequence
Dim lHistoryID As Long
Dim lRow As Long

Set oDB = CtxCreateObject("DawnOrder.DBordHist")
Set oSeq = CtxCreateObject("DawnSeq.BSequence")
lHistoryID = oSeq.GetNextInSequence("ord_hist")
lRow = oDB.AddOrderHistory(lHistoryID, alOrderID, alUsrID, alEmailInd,
asDescription, asEmailAddr, alEmailTrkCode, asIP, alOrderStatus)
If lRow <> 1 Then
    RaiseResError DawnOrder.DawnOrderErrors.ecAddOrderHistoryFailed,
TypeName(Me), "AddOrderHistory"
End If

Set oDB = Nothing
Set oSeq = Nothing
AddOrderHistory = lHistoryID
Exit Function

ErrHandler:
CtxSetAbort
Set oDB = Nothing
Exit Function

End Function

'=====
'this function adds the history for an item
'args - ord_id, item num, item status and other attributes for history
'returns item history ID
'=====
Private Function AddOrderItemHistory(ByVal alOrderID As Long, ByVal alItem As Long,
ByVal asDescription As String, _

```

```

                                BOrder.cls
ByVal asIP As String, ByVal alItemStatus As Long) As Long
On Error GoTo ErrHandler

    Dim oDB As DawnOrder.DBordItemHist
    Dim oSeq As DawnSeq.BSequence
    Dim lRow As Long
    Dim lHistoryID As Long

    Set oSeq = CtxCreateObject("DawnSeq.BSequence")
    Set oDB = CtxCreateObject("DawnOrder.DBordItemHist")

    lHistoryID = oSeq.GetNextInSequence("ord_item_hist")
    lRow = oDB.AddOrderItemHistory(lHistoryID, alOrderID, alItem, asDescription,
asEmailAddr, asIP, alItemStatus)
    If lRow <> 1 Then
        RaiseResError DawnOrder.DawnOrderErrors.ecAddOrderItemHistoryFailed,
TypeName(Me), "AddOrderItemHistory"
    End If

    Set oSeq = Nothing
    Set oDB = Nothing
    CtxSetComplete
    AddOrderItemHistory = lHistoryID
    Exit Function

ErrHandler:
    CtxSetAbort
    Set oDB = Nothing
    Set oSeq = Nothing
    Exit Function

End Function

```

```

                                dflt_tri_state_test.sql
if exists (select * from sysobjects
           where name = "sp_dflt_tri_state_test"
           and type = "p")
    drop procedure sp_dflt_tri_state_test
go

create procedure sp_dflt_tri_state_test ( @mfr_id int) as
create table #temp_memb (
    mfr_id int,
    prod_line_id int,
    commodity_id int,
    mfr_count int,
    avail_count int ,
    comm_ts_cd int,
    prod_ts_cd int)

insert into #temp_memb
select @mfr_id, prod_line_id, commodity_id , 0 , count(distinct part_id), 0 , 0
from memb_prod_ref
where mfr_id = @mfr_id
group by prod_line_id, commodity_id

update #temp_memb set mfr_count = (select count(distinct part_id)
from mfr_accept_prod_ref
where mfr_id = @mfr_id
and avail_ind = '1'
and accept_ind = '1'
and #temp_memb.prod_line_id = mfr_accept_prod_ref.prod_line_id
and #temp_memb.commodity_id = mfr_accept_prod_ref.commodity_id)

update #temp_memb set comm_ts_cd = case when avail_count = mfr_count then 1 when
avail_count > 0 then 2 else 0 end

update #temp_memb set avail_count = (select count(distinct part_id) from
memb_prod_ref m
where mfr_id = @mfr_id
and m.prod_line_id = #temp_memb.prod_line_id
and m.mfr_id = #temp_memb.mfr_id)

update #temp_memb set mfr_count = (select count(distinct part_id)
from mfr_accept_prod_ref
where mfr_id = @mfr_id
and avail_ind = '1'
and accept_ind = '1'
and #temp_memb.prod_line_id = mfr_accept_prod_ref.prod_line_id )

update #temp_memb set prod_ts_cd = case when avail_count = mfr_count then 1 when
avail_count > 0 then 2 else 0 end

update memb_prod_ref set comm_ts_cd = t.comm_ts_cd, prod_line_ts_cd = t.prod_ts_cd
from #temp_memb t
where memb_prod_ref.prod_line_id = t.prod_line_id
and memb_prod_ref.commodity_id = t.commodity_id
and memb_prod_ref.mfr_id = t.mfr_id

/*
select * from #temp_org

select t.org_id, t.prod_line_id, t.commodity_id, t.prod_ts_cd, r.prod_line_ts_cd,
t.comm_ts_cd , r.comm_ts_cd
from #temp_org t , org_prod_ref r
where t.org_id = r.org_id
and t.prod_line_id = r.prod_line_id

```

```
                                dflt_tri_state_test.sql
and t.commodity_id = r.commodity_id
and (t.prod_ts_cd <> r.prod_line_ts_cd or t.comm_ts_cd <> r.comm_ts_cd)

select count(*) from org_prod_ref where org_id = 2 and prod_line_id = 5
select count(*) from mfr_accept_prod_ref where mfr_id = 1 and prod_line_id = 5
*/
go

grant execute on sp_dflt_tri_state_test to public
go
```

```

                                mfr_dflt_price.sql
if exists (select * from sysobjects
           where name = "sp_mfr_dflt_price_calc_test"
           and type = "p")
    drop procedure sp_mfr_dflt_price_calc_test
go
create procedure sp_mfr_dflt_price_calc_test as
/*
create table #mfr_parts_add (
    mfr_id int,
    part_id int,
    commodity_id int,
    prod_line_id int
)

insert into #mfr_parts_add
    select distinct mfr_id, part_id, commodity_id, prod_line_id
    from memb_prod_ref
    where mfr_id = 2295 or mfr_id = 1199
*/
/**** create temp table *****/
--drop table #dflt_prod_temp
CREATE TABLE #dflt_prod_temp(
    mfr_id int NOT NULL ,
    part_id int NOT NULL ,
    prod_line_id int NOT NULL ,
    commodity_id int NOT NULL ,
    prod_line_ts_cd char (1) NOT NULL ,
    comm_ts_cd char (1) NOT NULL ,
    list_price money NULL ,
    sale_price money NULL ,
    special_price money NULL ,
    web_disc money NULL ,
    final_price money NULL ,
    disc_ind char (1) NOT NULL ,
    gbl_prod_line_disc money NULL ,
    gbl_prod_line_spec money NULL ,
    gbl_comm_disc money NULL ,
    gbl_comm_spec money NULL ,
    prod_line_web_disc money NULL ,
    prod_line_web_disc_typ_cd char (1) NULL ,
    comm_web_disc money NULL ,
    comm_web_disc_typ_cd char (1) NULL ,
    dflt_use_web_disc_ind char (1) NULL default 1
)

insert into #dflt_prod_temp
    (mfr_id,
    part_id,
    commodity_id,
    prod_line_id,
    prod_line_ts_cd,
    comm_ts_cd,
    list_price,
    sale_price,
    special_price,
    web_disc,
    final_price,
    disc_ind,
    gbl_prod_line_disc,
    gbl_prod_line_spec,
    gbl_comm_disc,
    gbl_comm_spec,
    prod_line_web_disc,

```

mfr_dflt_price.sql

```
prod_line_web_disc_typ_cd,  
comm_web_disc,  
comm_web_disc_typ_cd,  
dflt_use_web_disc_ind)
```

```
select  
m.mfr_id,  
m.part_id,  
m.commodity_id,  
m.prod_line_id,  
0,  
0,  
p.list_price,  
p.sale_price,  
NULL,  
NULL,  
0.00,  
0,  
0.00,  
0.00,  
0.00,  
0.00,  
pl.web_disc_val,  
pl.web_disc_typ_cd,  
pl.web_disc_val,  
p.web_disc_val,  
p.web_disc_typ_cd,
```

```
1  
from memb_prod_ref dp,  
mfr_comm_pricing p,  
mfr_prod_line_pricing pl,  
#mfr_parts_add m
```

```
where dp.mfr_id = m.mfr_id  
and dp.part_id = m.part_id  
and dp.commodity_id = m.commodity_id  
and dp.prod_line_id = m.prod_line_id  
and dp.mfr_id = p.mfr_id  
and dp.commodity_id = p.commodity_id  
and dp.mfr_id = pl.mfr_id  
and dp.prod_line_id = pl.prod_line_id
```

```
/*-----GLOBAL DISCOUNTS-----*/  
-- update global product line using percentage discount  
update #dflt_prod_temp  
set gbl_prod_line_disc = (select round(((disc_val/100)*  
#dflt_prod_temp.sale_price),2)  
from gbl_prod_line_disc a  
where mfr_id = #dflt_prod_temp.mfr_id  
and prod_line_id = #dflt_prod_temp.prod_line_id  
and disc_typ_cd = 1  
and disc_stus_cd = 1  
and disc_start_dte = (select max(disc_start_dte)  
from gbl_prod_line_disc b  
where mfr_id = a.mfr_id  
and prod_line_id = a.prod_line_id  
and disc_typ_cd = 1  
and disc_stus_cd = 1  
and disc_start_dte <=  
and disc_end_dte <= convert(datetime,convert(varchar(20),getdate(),1))  
and disc_end_dte is null
```

```
convert(datetime,convert(varchar(20),getdate(),1))  
)  
and disc_start_dte <= convert(datetime,convert(varchar(20),getdate(),1))  
and disc_end_dte is null
```

```

                                mfr_dflt_price.sql
where exists(select 'x'
from gbl_prod_line_disc c
where mfr_id      = #dflt_prod_temp.mfr_id
  and prod_line_id = #dflt_prod_temp.prod_line_id
  and disc_typ_cd  = 1
  and disc_stus_cd = 1
  and disc_start_dte = (select max(disc_start_dte)
                        from gbl_prod_line_disc d
                        where mfr_id      = c.mfr_id
                          and prod_line_id = c.prod_line_id
                          and disc_typ_cd  = 1
                          and disc_stus_cd = 1
                          and disc_start_dte <=
convert(datetime,convert(varchar(20),getdate(),1))
                        and disc_end_dte is null
                        )
  and disc_end_dte is null
)

-- update global product line using percentage special
update #dflt_prod_temp
set gbl_prod_line_spec = (select round(((disc_val/100)*
#dflt_prod_temp.sale_price),2)
from gbl_prod_line_disc a
where mfr_id      = #dflt_prod_temp.mfr_id
  and prod_line_id = #dflt_prod_temp.prod_line_id
  and disc_typ_cd  = 1
  and disc_stus_cd = 1
  and disc_start_dte = (select max(disc_start_dte)
                        from gbl_prod_line_disc b
                        where mfr_id      = a.mfr_id
                          and prod_line_id = a.prod_line_id
                          and disc_typ_cd  = 1
                          and disc_stus_cd = 1
                          and disc_start_dte <=
convert(datetime,convert(varchar(20),getdate(),1))
                        and disc_end_dte >=
convert(datetime,convert(varchar(20),getdate(),1))
                        )
  and disc_start_dte <= convert(datetime,convert(varchar(20),getdate(),1))
  and disc_end_dte >= convert(datetime,convert(varchar(20),getdate(),1))
)

where exists(select 'x'
from gbl_prod_line_disc c
where mfr_id      = #dflt_prod_temp.mfr_id
  and prod_line_id = #dflt_prod_temp.prod_line_id
  and disc_typ_cd  = 1
  and disc_stus_cd = 1
  and disc_start_dte = (select max(disc_start_dte)
                        from gbl_prod_line_disc d
                        where mfr_id      = c.mfr_id
                          and prod_line_id = c.prod_line_id
                          and disc_typ_cd  = 1
                          and disc_stus_cd = 1
                          and disc_start_dte <=
convert(datetime,convert(varchar(20),getdate(),1))
                        and disc_end_dte >=
convert(datetime,convert(varchar(20),getdate(),1))
                        )
  and disc_end_dte >= convert(datetime,convert(varchar(20),getdate(),1))
)

-- update global product line using dollar discount

```



```

                                mfr_dflt_price.sql
update #dflt_prod_temp
set gbl_prod_line_disc = (select disc_val
  from gbl_prod_line_disc a
 where mfr_id           = #dflt_prod_temp.mfr_id
 and prod_line_id      = #dflt_prod_temp.prod_line_id
 and disc_typ_cd       = 2
 and disc_stus_cd      = 1
 and disc_start_dte = (select max(disc_start_dte)
                       from gbl_prod_line_disc b
                       where mfr_id           = a.mfr_id
                         and prod_line_id      = a.prod_line_id
                         and disc_typ_cd       = 2
                         and disc_stus_cd      = 1
                         and disc_start_dte <=
convert(datetime,convert(varchar(20),getdate(),1))
                        and disc_end_dte is null
                       )
      and disc_start_dte <= convert(datetime,convert(varchar(20),getdate(),1))
      and disc_end_dte is null
)
where exists(select 'x'
  from gbl_prod_line_disc c
 where mfr_id           = #dflt_prod_temp.mfr_id
 and prod_line_id      = #dflt_prod_temp.prod_line_id
 and disc_typ_cd       = 2
 and disc_stus_cd      = 1
 and disc_start_dte = (select max(disc_start_dte)
                       from gbl_prod_line_disc d
                       where mfr_id           = c.mfr_id
                         and prod_line_id      = c.prod_line_id
                         and disc_typ_cd       = 2
                         and disc_stus_cd      = 1
                         and disc_start_dte <=
convert(datetime,convert(varchar(20),getdate(),1))
                        and disc_end_dte is null
                       )
      and disc_end_dte is null
)

-- update global product line using dollar special
update #dflt_prod_temp
set gbl_prod_line_spec = (select disc_val
  from gbl_prod_line_disc a
 where mfr_id           = #dflt_prod_temp.mfr_id
 and prod_line_id      = #dflt_prod_temp.prod_line_id
 and disc_typ_cd       = 2
 and disc_stus_cd      = 1
 and disc_start_dte = (select max(disc_start_dte)
                       from gbl_prod_line_disc b
                       where mfr_id           = a.mfr_id
                         and prod_line_id      = a.prod_line_id
                         and disc_typ_cd       = 2
                         and disc_stus_cd      = 1
                         and disc_start_dte <=
convert(datetime,convert(varchar(20),getdate(),1))
                        and disc_end_dte >=
convert(datetime,convert(varchar(20),getdate(),1))
                       )
      and disc_start_dte <= convert(datetime,convert(varchar(20),getdate(),1))
      and disc_end_dte >= convert(datetime,convert(varchar(20),getdate(),1))
)
where exists(select 'x'

```

```

                                mfr_dflt_price.sql
from gbl_prod_line_disc c
where mfr_id      = #dflt_prod_temp.mfr_id
   and prod_line_id = #dflt_prod_temp.prod_line_id
   and disc_typ_cd = 2
   and disc_stus_cd = 1
   and disc_start_dte = (select max(disc_start_dte)
                        from gbl_prod_line_disc d
                        where mfr_id      = c.mfr_id
                           and prod_line_id = c.prod_line_id
                           and disc_typ_cd = 2
                           and disc_stus_cd = 1
                           and disc_start_dte <=
convert(datetime,convert(varchar(20),getdate(),1))
                           and disc_end_dte >=
convert(datetime,convert(varchar(20),getdate(),1))
                        )
   and disc_end_dte >= convert(datetime,convert(varchar(20),getdate(),1))
)

-- update global commodity using percentage discount
update #dflt_prod_temp
set gbl_comm_disc = (select round(((disc_val/100)* #dflt_prod_temp.sale_price),2)
                    from gbl_comm_disc a
                    where mfr_id      = #dflt_prod_temp.mfr_id
                       and commodity_id = #dflt_prod_temp.commodity_id
                       and disc_typ_cd = 1
                       and disc_stus_cd = 1
                       and disc_start_dte = (select max(disc_start_dte)
                                             from gbl_comm_disc b
                                             where mfr_id      = a.mfr_id
                                                and commodity_id = a.commodity_id
                                                and disc_typ_cd = 1
                                                and disc_stus_cd = 1
                                                and disc_start_dte <=
convert(datetime,convert(varchar(20),getdate(),1))
                                                and disc_end_dte is null
                                             )
                       and disc_start_dte <= convert(datetime,convert(varchar(20),getdate(),1))
                       and disc_end_dte is null
                    )
where exists(select 'x'
            from gbl_comm_disc c
            where mfr_id      = #dflt_prod_temp.mfr_id
               and commodity_id = #dflt_prod_temp.commodity_id
               and disc_typ_cd = 1
               and disc_stus_cd = 1
               and disc_start_dte = (select max(disc_start_dte)
                                     from gbl_comm_disc d
                                     where mfr_id      = c.mfr_id
                                        and commodity_id = c.commodity_id
                                        and disc_typ_cd = 1
                                        and disc_stus_cd = 1
                                        and disc_start_dte <=
convert(datetime,convert(varchar(20),getdate(),1))
                                        and disc_end_dte is null
                                     )
               and disc_end_dte is null
            )

-- update global commodity using percentage special
update #dflt_prod_temp
set gbl_comm_spec = (select round(((disc_val/100)* #dflt_prod_temp.sale_price),2)

```

```

                                mfr_dflt_price.sql
from gbl_comm_disc a
where mfr_id      = #dflt_prod_temp.mfr_id
and commodity_id = #dflt_prod_temp.commodity_id
and disc_typ_cd  = 1
and disc_stus_cd = 1
and disc_start_dte = (select max(disc_start_dte)

                                from gbl_comm_disc b
                                where mfr_id      = a.mfr_id
                                   and commodity_id = a.commodity_id
                                   and disc_typ_cd  = 1
                                   and disc_stus_cd = 1
                                   and disc_start_dte <=
convert(datetime,convert(varchar(20),getdate(),1))
                                   and disc_end_dte >=
convert(datetime,convert(varchar(20),getdate(),1))
                                )
and disc_start_dte <= convert(datetime,convert(varchar(20),getdate(),1))
and disc_end_dte >= convert(datetime,convert(varchar(20),getdate(),1))
)
where exists(select 'x'
from gbl_comm_disc c
where mfr_id      = #dflt_prod_temp.mfr_id
and commodity_id = #dflt_prod_temp.commodity_id
and disc_typ_cd  = 1
and disc_stus_cd = 1
and disc_start_dte = (select max(disc_start_dte)
from gbl_comm_disc d
where mfr_id      = c.mfr_id
and commodity_id = c.commodity_id
and disc_typ_cd  = 1
and disc_stus_cd = 1
and disc_start_dte <=
convert(datetime,convert(varchar(20),getdate(),1))
and disc_end_dte >=
convert(datetime,convert(varchar(20),getdate(),1))
)
and disc_end_dte >= convert(datetime,convert(varchar(20),getdate(),1))
)

-- update global commodity using dollar discount
update #dflt_prod_temp
set gbl_comm_disc = (select disc_val
from gbl_comm_disc a
where mfr_id      = #dflt_prod_temp.mfr_id
and commodity_id = #dflt_prod_temp.commodity_id
and disc_typ_cd  = 2
and disc_stus_cd = 1
and disc_start_dte = (select max(disc_start_dte)
from gbl_comm_disc b
where mfr_id      = a.mfr_id
and commodity_id = a.commodity_id
and disc_typ_cd  = 2
and disc_stus_cd = 1
and disc_start_dte <=
convert(datetime,convert(varchar(20),getdate(),1))
and disc_end_dte is null
)
and disc_start_dte <= convert(datetime,convert(varchar(20),getdate(),1))
and disc_end_dte is null
)
where exists(select 'x'

```

```

                                mfr_dflt_price.sql
from gbl_comm_disc c
where mfr_id = #dflt_prod_temp.mfr_id
   and commodity_id = #dflt_prod_temp.commodity_id
   and disc_typ_cd = 2
   and disc_stus_cd = 1
   and disc_start_dte = (select max(disc_start_dte)
                        from gbl_comm_disc d
                        where mfr_id = c.mfr_id
                           and commodity_id = c.commodity_id
                           and disc_typ_cd = 2
                           and disc_stus_cd = 1
                           and disc_start_dte <=
convert(datetime,convert(varchar(20),getdate(),1))
                        and disc_end_dte is null
                        )
   and disc_end_dte is null
)

-- update global commodity using dollar special
update #dflt_prod_temp
set gbl_comm_spec = (select disc_val
                    from gbl_comm_disc a
                    where mfr_id = #dflt_prod_temp.mfr_id
                       and commodity_id = #dflt_prod_temp.commodity_id
                       and disc_typ_cd = 2
                       and disc_stus_cd = 1
                       and disc_start_dte = (select max(disc_start_dte)
                                           from gbl_comm_disc b
                                           where mfr_id = a.mfr_id
                                              and commodity_id = a.commodity_id
                                              and disc_typ_cd = 2
                                              and disc_stus_cd = 1
                                              and disc_start_dte <=
convert(datetime,convert(varchar(20),getdate(),1))
                                           and disc_end_dte >=
convert(datetime,convert(varchar(20),getdate(),1))
                                           )
                       and disc_start_dte <= convert(datetime,convert(varchar(20),getdate(),1))
                       and disc_end_dte >= convert(datetime,convert(varchar(20),getdate(),1))
                    )
where exists(select 'x'
            from gbl_comm_disc c
            where mfr_id = #dflt_prod_temp.mfr_id
               and commodity_id = #dflt_prod_temp.commodity_id
               and disc_typ_cd = 2
               and disc_stus_cd = 1
               and disc_start_dte = (select max(disc_start_dte)
                                    from gbl_comm_disc d
                                    where mfr_id = c.mfr_id
                                       and commodity_id = c.commodity_id
                                       and disc_typ_cd = 2
                                       and disc_stus_cd = 1
                                       and disc_start_dte <=
convert(datetime,convert(varchar(20),getdate(),1))
                                       and disc_end_dte >=
convert(datetime,convert(varchar(20),getdate(),1))
                                       )
               and disc_end_dte >= convert(datetime,convert(varchar(20),getdate(),1))
            )

/***** Determine Largest Discount or Special *****/
/****specials****/

```

mfr_dflt_price.sql

```

update #dflt_prod_temp
set special_price = sale_price - round(gbl_prod_line_spec,2)
where gbl_prod_line_spec > gbl_comm_spec

update #dflt_prod_temp
set special_price = sale_price - round(gbl_comm_spec,2)
where gbl_comm_spec > gbl_prod_line_spec

/*****discount*****/
update #dflt_prod_temp
set sale_price = sale_price - round(gbl_prod_line_disc,2),
    disc_ind = 1
where gbl_prod_line_disc > gbl_comm_disc

update #dflt_prod_temp
set sale_price = sale_price - round(gbl_comm_disc,2),
    disc_ind = 1
where gbl_comm_disc > gbl_prod_line_disc

/*** update the final_price with the sale price ****/
update #dflt_prod_temp
set final_price = round(sale_price,2)
where 1 = 1

update #dflt_prod_temp
set final_price = -9999999.9900
where final_price is null

update #dflt_prod_temp
set sale_price = 0.00
where sale_price is null

update #dflt_prod_temp
set list_price = 0.00
where list_price is null

/*** overwrite the final_price if the special prices is lower ****/
update #dflt_prod_temp
set final_price = round(special_price,2)
where sale_price > special_price
and special_price is not NULL

/***** translate the percentage web discounts*****/
update #dflt_prod_temp
set comm_web_disc = (final_price*(comm_web_disc/100))
where comm_web_disc_typ_cd = 1
    and comm_web_disc > 0

update #dflt_prod_temp
set prod_line_web_disc = (final_price*(prod_line_web_disc/100))
where prod_line_web_disc_typ_cd = 1
    and prod_line_web_disc > 0

/***** determine the greatest web discount*****/
update #dflt_prod_temp
set web_disc = round(prod_line_web_disc,2)
where prod_line_web_disc <> 0
    and prod_line_web_disc is not NULL
    and prod_line_web_disc >= comm_web_disc

update #dflt_prod_temp
set web_disc = round(comm_web_disc,2)

```

```

                                mfr_dflt_price.sql
where prod_line_web_disc <> 0
   and prod_line_web_disc is not NULL
   and comm_web_disc >= prod_line_web_disc

/***** subtract the web discount from the final price if web_use_disc_ind is 1
*****/
update #dflt_prod_temp
set final_price = final_price-web_disc
where web_disc <> 0
   and web_disc is not NULL
   and final_price > 0
   and dflt_use_web_disc_ind = 1

-- update org_prod_ref with the price structure
update memb_prod_ref set
list_price = t.list_price,
sale_price = t.sale_price,
special_price = t.special_price,
final_price = t.final_price,
web_disc = t.web_disc,
disc_ind = t.disc_ind
from #dflt_prod_temp t
where memb_prod_ref.mfr_id = t.mfr_id
and memb_prod_ref.part_id = t.part_id
and memb_prod_ref.commodity_id = t.commodity_id
and memb_prod_ref.prod_line_id = t.prod_line_id

go

grant execute on sp_mfr_dflt_price_calc_test to public
go

```

```

                                org_tri_state_test.sql
if exists (select * from sysobjects
           where name = "sp_org_tri_state_test"
           and type = "p")
    drop procedure sp_org_tri_state_test
go

create procedure sp_org_tri_state_test ( @mfr_id int) as
create table #temp_org (
    org_id int,
    prod_line_id int,
    commodity_id int,
    mfr_count int,
    org_count int,
    comm_ts_cd int,
    prod_ts_cd int)

insert into #temp_org
select org.org_id, prod_line_id, commodity_id , 0 , count(distinct part_id), 0 , 0
from org_prod_ref, org
where org.mfr_id = @mfr_id
and org_prod_ref.org_id = org.org_id
group by org.org_id, prod_line_id, commodity_id

update #temp_org set mfr_count = (select count(distinct part_id)
from mfr_accept_prod_ref
where mfr_id = @mfr_id
and avail_ind = '1'
and accept_ind = '1'
and #temp_org.prod_line_id = mfr_accept_prod_ref.prod_line_id
and #temp_org.commodity_id = mfr_accept_prod_ref.commodity_id)

update #temp_org set comm_ts_cd = case when org_count = mfr_count then 1 else 2 end

update #temp_org set org_count = (select count(distinct part_id) from org_prod_ref,
org
where org.mfr_id = @mfr_id
and org_prod_ref.org_id = org.org_id
and org_prod_ref.prod_line_id = #temp_org.prod_line_id
and org_prod_ref.org_id = #temp_org.org_id)

update #temp_org set mfr_count = (select count(distinct part_id)
from mfr_accept_prod_ref
where mfr_id = @mfr_id
and avail_ind = '1'
and accept_ind = '1'
and #temp_org.prod_line_id = mfr_accept_prod_ref.prod_line_id )

update #temp_org set prod_ts_cd = case when org_count = mfr_count then 1 else 2 end

update org_prod_ref set comm_ts_cd = t.comm_ts_cd, prod_line_ts_cd = t.prod_ts_cd
from #temp_org t
where org_prod_ref.org_id = t.org_id
and org_prod_ref.prod_line_id = t.prod_line_id
and org_prod_ref.commodity_id = t.commodity_id

/*
select * from #temp_org

select t.org_id, t.prod_line_id, t.commodity_id, t.prod_ts_cd, r.prod_line_ts_cd,
t.comm_ts_cd , r.comm_ts_cd
from #temp_org t , org_prod_ref r
where t.org_id = r.org_id

```

```
org_tri_state_test.sql
and t.prod_line_id = r.prod_line_id
and t.commodity_id = r.commodity_id
and (t.prod_ts_cd <> r.prod_line_ts_cd or t.comm_ts_cd <> r.comm_ts_cd)

select count(*) from org_prod_ref where org_id = 2 and prod_line_id = 5
select count(*) from mfr_accept_prod_ref where mfr_id = 1 and prod_line_id = 5
*/

go

grant execute on sp_org_tri_state_test to public
go
```



```

sp_mfr_accept_tri_state_test.sql
if exists (select * from sysobjects
           where name = "sp_mfr_accept_tri_state_test"
           and type = "p")
    drop procedure sp_mfr_accept_tri_state_test
go

create procedure sp_mfr_accept_tri_state_test ( @mfr_id int, @src_mfr_id int) as
-- new SP for mfr accept tri states for the whole src

--2 args - mfr ID and src mfr ID
--drop table #tri_state

create table #tri_state (
    mfr_id int,
    src_mfr_id int,
    prod_line_id int,
    commodity_id int,
    part_avail int,
    part_accept int,
    comm_ts_cd int,
    prod_ts_cd int,
    mfr_ts_cd int )

insert into #tri_state
select @mfr_id, @src_mfr_id, prod_line_id, commodity_id, 0, 0, 0, 0, 0
from mfr_accept_part
where mfr_id = @mfr_id
and src_mfr_id = @src_mfr_id
order by prod_line_id, commodity_id

update #tri_state
set part_avail = (select count(m.part_id)
from mfr_accept_part m
where #tri_state.mfr_id = m.mfr_id
and #tri_state.src_mfr_id = m.src_mfr_id
and #tri_state.prod_line_id = m.prod_line_id
and #tri_state.commodity_id = m.commodity_id
and m.avail_ind = '1')

update #tri_state
set part_accept = (select count(m.part_id)
from mfr_accept_part m
where #tri_state.mfr_id = m.mfr_id
and #tri_state.src_mfr_id = m.src_mfr_id
and #tri_state.prod_line_id = m.prod_line_id
and #tri_state.commodity_id = m.commodity_id
and m.accept_ind = '1' and m.avail_ind = '1')

update #tri_state set comm_ts_cd = case when ( part_avail = part_accept and
part_avail > 0 ) then 1 when part_accept > 0 then 2 else 0 end

update #tri_state
set part_avail = (select count(m.part_id)
from mfr_accept_part m
where #tri_state.mfr_id = m.mfr_id
and #tri_state.src_mfr_id = m.src_mfr_id
and #tri_state.prod_line_id = m.prod_line_id
and m.avail_ind = '1')

update #tri_state
set part_accept = ( select count(m.part_id)
from mfr_accept_part m
where #tri_state.mfr_id = m.mfr_id

```

```

        sp_mfr_accept_tri_state_test.sql
        and #tri_state.src_mfr_id = m.src_mfr_id
        and #tri_state.prod_line_id = m.prod_line_id
        and m.accept_ind = '1' and m.avail_ind = '1' )

update #tri_state set prod_ts_cd = case when ( part_avail = part_accept and
part_avail > 0 ) then 1 when part_accept > 0 then 2 else 0 end

update #tri_state
    set part_avail = (select count(m.part_id)
    from mfr_accept_part m
    where #tri_state.mfr_id = m.mfr_id
    and #tri_state.src_mfr_id = m.src_mfr_id
    and m.avail_ind = '1')

update #tri_state
    set part_accept = ( select count(m.part_id)
    from mfr_accept_part m
    where #tri_state.mfr_id = m.mfr_id
    and #tri_state.src_mfr_id = m.src_mfr_id
    and m.accept_ind = '1' and m.avail_ind = '1' )

update #tri_state set mfr_ts_cd = case when ( part_avail = part_accept and
part_avail > 0 ) then 1 when part_accept > 0 then 2 else 0 end

update mfr_accept_prod_ref
    set comm_ts_cd = t.comm_ts_cd
    from #tri_state t
    where t.mfr_id = mfr_accept_prod_ref.mfr_id
    and t.src_mfr_id = mfr_accept_prod_ref.src_mfr_id
    and t.commodity_id = mfr_accept_prod_ref.commodity_id
    and t.prod_line_id = mfr_accept_prod_ref.prod_line_id

update mfr_accept_prod_ref
    set prod_line_ts_cd = t.prod_ts_cd
    from #tri_state t
    where t.mfr_id = mfr_accept_prod_ref.mfr_id
    and t.src_mfr_id = mfr_accept_prod_ref.src_mfr_id
    and t.commodity_id = mfr_accept_prod_ref.commodity_id
    and t.prod_line_id = mfr_accept_prod_ref.prod_line_id

update mfr_accept_prod_ref
    set mfr_ts_cd = t.mfr_ts_cd
    from #tri_state t
    where t.mfr_id = mfr_accept_prod_ref.mfr_id
    and t.src_mfr_id = mfr_accept_prod_ref.src_mfr_id
    and t.commodity_id = mfr_accept_prod_ref.commodity_id
    and t.prod_line_id = mfr_accept_prod_ref.prod_line_id

/*
select * from #tri_state

select mfr_id, src_mfr_id, prod_line_id, commodity_id, part_id, prod_line_ts_cd,
comm_ts_cd
from mfr_accept_prod_ref
where mfr_id = 1 and src_mfr_id = 0
order by prod_line_id, commodity_id, part_id

select t.prod_line_id, t.commodity_id, t.prod_ts_cd, m.prod_line_ts_cd, t.comm_ts_cd
, m.comm_ts_cd
    from #tri_state t, mfr_accept_prod_ref m
    where t.mfr_id = m.mfr_id
    and t.src_mfr_id = m.src_mfr_id
    and t.prod_line_id = m.prod_line_id

```

```
sp_mfr_accept_tri_state_test.sql
and t.commodity_id *= m.commodity_id
and t.part_id *= m.part_id
*/
go

grant execute on sp_mfr_accept_tri_state_test to public
go
```

```

SET QUOTED_IDENTIFIER OFF      sp_pipeline_mfr_add.sql
SET ANSI_NULLS ON
GO

if exists (select * from sysobjects where id =
object_id(N'[dbo].[sp_pipeline_mfr_add]') and OBJECTPROPERTY(id, N'IsProcedure') =
1)
drop procedure [dbo].[sp_pipeline_mfr_add]
GO

/*
* sp_pipeline_mfr_add
* this procedure will set avail = 1 for all pipeline orgs for one mfr
* uday 15 feb 2000
*/

CREATE PROCEDURE sp_pipeline_mfr_add ( @lSrcMfr integer) AS

declare @lOrg integer

-- create temp table
create table #templ( org_id integer primary key)

-- get all pipeline orgs for this mfr
insert into #templ select distinct org_id from org where mfr_id = @lSrcMfr and
pl_ind = '1' and pl_mfr_id is not null

-- loop thru a cursor and set all the orgs avail ind = 0
declare pipe_org cursor for select org_id from #templ
open pipe_org
fetch next from pipe_org into @lOrg
while @@fetch_status = 0
begin
    exec sp_pipeline_org_add @lOrg , 0
    fetch next from pipe_org into @lOrg
end
close pipe_org
deallocate pipe_org

GO
SET QUOTED_IDENTIFIER OFF      SET ANSI_NULLS ON
GO

GRANT EXECUTE ON [dbo].[sp_pipeline_mfr_add] TO [public]
GO

```

```

1
      sp_pipeline_mfr_product_add.sql.txt
SET QUOTED_IDENTIFIER OFF      SET ANSI_NULLS ON
GO

if exists (select * from sysobjects where id =
object_id(N'[dbo].[sp_pipeline_mfr_product_add]') and OBJECTPROPERTY(id,
N'IsProcedure') = 1)
drop procedure [dbo].[sp_pipeline_mfr_product_add]
GO

/*
sp_pipeline_mfr_product_add
this procedure removed a prod for all the pipeline orgs for this mfr
this can happen when a supplier makes the prod unavaliable to the source mfr or if
the source mfr unaccepts the prod
*/

CREATE PROCEDURE sp_pipeline_mfr_product_add ( @lSrcMfr int , @lProdline int = -1 ,
@lCommodity int = -1 , @lPart int = -1 ) AS

declare @lOrg integer

-- check to make sure at least 1 ID has been passed
if @lProdline = -1 and @lCommodity = 1 and @lPart = -1
begin
    return
end

-- create temp table
create table #temp1 ( org_id int primary key)

-- get all pipeline orgs
insert into #temp1 select distinct org_id from org where mfr_id = @lSrcMfr and
pl_ind = '1' and pl_mfr_id is not null

-- loop thru and call the org SP to do the actual job
declare pipe_org cursor for select org_id from #temp1
open pipe_org
fetch next from pipe_org into @lOrg
while @@fetch_status = 0
begin
    exec sp_pipeline_org_product_add @lOrg , @lProdLine , @lCommodity , @lPart ,
0
    fetch next from pipe_org into @lOrg
end
close pipe_org
deallocate pipe_org

GO
SET QUOTED_IDENTIFIER OFF      SET ANSI_NULLS ON
GO

GRANT EXECUTE ON [dbo].[sp_pipeline_mfr_product_add] TO [public]
GO

```

```

SET QUOTED_IDENTIFIER OFF SET ANSI_NULLS ON
GO

```

```

if exists (select * from sysobjects where id =
object_id(N'[dbo].[sp_pipeline_mfr_product_remove]') and OBJECTPROPERTY(id,
N'IsProcedure') = 1)
drop procedure [dbo].[sp_pipeline_mfr_product_remove]
GO

```

```

/*
sp_pipeline_mfr_product_remove
this procedure removed a prod for all the pipeline orgs for this mfr
this can happen when a supplier makes the prod unavailable to the source mfr or if
the source mfr unaccepts the prod
*/

```

```

CREATE PROCEDURE sp_pipeline_mfr_product_remove ( @lSrcMfr int , @lProdline int =
-1 , @lCommodity int = -1 , @lPart int = -1 ) AS

```

```

declare @lOrg integer

```

```

-- check to make sure at least 1 ID has been passed
if @lProdline = -1 and @lCommodity = 1 and @lPart = -1
begin
    return
end

```

```

-- create temp table
create table #temp1 ( org_id int primary key)

```

```

-- get all pipeline orgs
insert into #temp1 select distinct org_id from org where mfr_id = @lSrcMfr and
pl_ind = '1' and pl_mfr_id is not null

```

```

-- loop thru and call the org SP to do the actual job
declare pipe_org cursor for select org_id from #temp1
open pipe_org
fetch next from pipe_org into @lOrg
while @@fetch_status = 0
begin
    exec sp_pipeline_org_product_remove @lOrg , @lProdLine , @lCommodity ,
    @lPart
    fetch next from pipe_org into @lOrg
end
close pipe_org
deallocate pipe_org

```

```

GO
SET QUOTED_IDENTIFIER OFF SET ANSI_NULLS ON
GO

```

```

GRANT EXECUTE ON [dbo].[sp_pipeline_mfr_product_remove] TO [public]
GO

```

```

SET QUOTED_IDENTIFIER OFF      sp_pipeline_mfr_remove.sql
SET ANSI_NULLS ON
GO

if exists (select * from sysobjects where id =
object_id(N'[dbo].[sp_pipeline_mfr_remove]') and OBJECTPROPERTY(id, N'IsProcedure')
= 1)
drop procedure [dbo].[sp_pipeline_mfr_remove]
GO

/*
* sp_pipeline_mfr_remove
* this procedure will set avail = 0 for all pipeline orgs for one mfr
* uday 15 feb 2000
*/

CREATE PROCEDURE sp_pipeline_mfr_remove ( @lSrcMfr integer) AS

declare @lOrg integer

-- create temp table
create table #temp1( org_id integer primary key)

-- get all pipeline orgs for this mfr
insert into #temp1 select distinct org_id from org where mfr_id = @lSrcMfr and
pl_ind = '1' and pl_mfr_id is not null

-- loop thru a cursor and set all the orgs avail ind = 0
declare pipe_org cursor for select org_id from #temp1
open pipe_org
fetch next from pipe_org into @lOrg
while @@fetch_status = 0
begin
    exec sp_pipeline_org_remove @lOrg , 0
    fetch next from pipe_org into @lOrg
end
close pipe_org
deallocate pipe_org
GO
SET QUOTED_IDENTIFIER OFF      SET ANSI_NULLS ON
GO

GRANT EXECUTE ON [dbo].[sp_pipeline_mfr_remove] TO [public]
GO

```

```

SET QUOTED_IDENTIFIER OFF      sp_pipeline_org_add.sql
SET ANSI_NULLS ON
GO

```

```

if exists (select * from sysobjects where id =
object_id(N'[dbo].[sp_pipeline_org_add]') and OBJECTPROPERTY(id, N'IsProcedure') =
1)
drop procedure [dbo].[sp_pipeline_org_add]
GO

```

```

/*
* sp_pipeline_org_add
* this procedure will subscribe all parts in a pipeline org to the pipeline mfr
* lNew is a indicator to let this SP know if new rows are added or if old status is
being restored
* uday jan 31 2000
*/

```

```

CREATE PROCEDURE sp_pipeline_org_add ( @lPipelineOrg int , @lNew int = 1 )
as

```

```

declare @lSrcMfr integer
declare @lPipelineMfr integer
declare @lProd integer
declare @lOrg integer

```

```

-- get mfr IDs for src and pipeline
select @lSrcMfr = mfr_id from org where org_id = @lPipelineOrg
select @lPipelineMfr = pl_mfr_id from org where org_id = @lPipelineOrg

```

```

-- check for pipeline mfr
if ( @lPipelineMfr is null or @lPipelineMfr <= 0 )
begin

```

```

    return
end

```

```

-- create temp table
create table #temp1 ( org_id int primary key)

```

```

--if rows already exist in the mfr accept part table , then turn on the avail ind
else insert all the rows

```

```

-- else get all rows from the org prod ref table and add it to the
--mfr_accept_part table with src_mfr as @lsrcmfr and mfr_id as @lpipelinemfr
if exists ( select part_id from mfr_accept_part where mfr_id = @lPipelineMfr and
src_mfr_id = @lSrcMfr)
begin

```

```

    update mfr_accept_part set avail_ind = '1'
    where mfr_id = @lPipelineMfr and src_mfr_id = @lSrcMfr

```

```

    if @lNew = 1
    begin

```

```

        -- if any of the parts here have been accpeted and been pipelined to
other orgs, then those orgs should also have avail ind = 1

```

```

        insert into #temp1 select distinct org_prod_ref.org_id from
org_prod_ref, org

```

```

        where mfr_id = @lPipelineMfr
        and part_id not in ( select distinct part_id from mfr_accept_part
where mfr_id = @lPipelineMfr and src_mfr_id <> @lSrcMfr and accept_ind = '1')
        and org.org_id = org_prod_ref.org_id and org.pl_ind = '1'

```

```

    end

```

```

end
else
begin

```

```

    insert into mfr_accept_part ( mfr_id, part_id, src_mfr_id, prod_line_id,
Page 1

```



```

                                sp_pipeline_org_add.sql
commodity_id, accept_ind, dflt_src_ind, avail_ind, create_dtm,
last_maint_dtm )
select @lPipelineMfr, part_id, @lSrcMfr , prod_line_id, commodity_id, '0' ,
'0' , '1' , getdate(), getdate()
from org_prod_ref where org_id = @lPipelineOrg
end

-- if there are any dependent pipeline orgs, then set avail_ind = 1 for all those
orgs too
if (select count(*) from #temp1) > 0
begin
    declare pipe_org cursor for select org_id from #temp1
    open pipe_org
    fetch next from pipe_org into @lOrg
    while @@fetch_status = 0
    begin
        exec sp_pipeline_org_add @lOrg
        fetch next from pipe_org into @lOrg
    end
    close pipe_org
    deallocate pipe_org
end

GO
SET QUOTED_IDENTIFIER OFF      SET ANSI_NULLS ON
GO

GRANT EXECUTE ON [dbo].[sp_pipeline_org_add] TO [public]
GO

```

```

SET QUOTED_IDENTIFIER OFF SET ANSI_NULLS ON
GO
sp_pipeline_org_product_add.sql
GO
if exists (select * from sysobjects where id =
object_id(N'[dbo].[sp_pipeline_org_product_add]') and OBJECTPROPERTY(id,
N'IsProcedure') = 1)
drop procedure [dbo].[sp_pipeline_org_product_add]
GO
/*
* sp_pipeline_org_product_add
* this procedure will subscribe all parts in a pipeline org to the pipeline mfr
* lNew is a indicator to let this SP know if new rows will be added or just the old
status will be restored for this pipeline org
* uday jan 31 2000
*/
CREATE PROCEDURE sp_pipeline_org_product_add ( @lPipelineOrg int , @lProdLine int =
-1 , @lCommodity int = -1 , @lPart int = -1 , @lNew int = 1)
as
declare @lSrcMfr integer
declare @lPipelineMfr integer
declare @lLocalPart integer
declare @lLocalOrg integer

-- get mfr IDs for src and pipeline
select @lSrcMfr = mfr_id from org where org_id = @lPipelineOrg
select @lPipelineMfr = pl_mfr_id from org where org_id = @lPipelineOrg

-- check for pipeline mfr
if ( @lPipelineMfr is null or @lPipelineMfr <= 0 )
begin
    return
end
--create temp tables temp1 for dflt prod , temp2 for org prod and temp3 for
recursive pipeline orgs
create table #temp1 (part_id int primary key)
create table #temp2 (org_id int not null , part_id int not null )
alter table #temp2 add constraint pk_temp2 primary key (org_id, part_id)
create table #temp3 (org_id int not null , part_id int not null )
alter table #temp3 add constraint pk_temp3 primary key (org_id, part_id)

--check for error
if @lProdLine = -1 and @lCommodity = -1 and @lPart = -1
begin
    return
--
    raiserror ('The product ID supplied is not valid.' , 1 , 1)
end

-- if prodline is being added, then insert all parts into the mfr_accept_part table
except for the parts which are already in there.
-- for these parts, turn avail_ind = 1

if @lProdLine > 0
begin
    -- update already existing parts
    update mfr_accept_part set avail_ind = '1'
    where mfr_id = @lPipelineMfr and src_mfr_id = @lSrcMfr and prod_line_id =
@lProdLine

    if @lNew = 1

```

```

                                sp_pipeline_org_product_add.sql
begin
    -- insert new parts for this prod line
    insert into mfr_accept_part ( mfr_id, part_id, src_mfr_id,
prod_line_id,
    commodity_id, accept_ind, dflt_src_ind, avail_ind, create_dtm,
last_maint_dtm )
    select @lPipelineMfr, part_id, @lSrcMfr , prod_line_id,
commodity_id, '0'
    '0', '1' , getdate(), getdate()
    from org_prod_ref where org_id = @lPipelineOrg and prod_line_id =
@lProdLine
    and part_id not in (select distinct part_id from mfr_accept_part
    where mfr_id = @lPipelineMfr and src_mfr_id = @lSrcMfr and
prod_line_id = @lProdLine)
    end
/*
    -- get list of all parts with access to dflt members, the memb prod ref
table needs to be updated for these parts
    insert into #temp1 select distinct part_id from dflt_avail_part
    where part_id in (select distinct part_id from mfr_accept_part
    where mfr_id = @lPipelineMfr and src_mfr_id = @lSrcMfr and prod_line_id =
@lProdLine)
    and part_id not in (select distinct part_id from mfr_accept_part
    where mfr_id = @lPipelineMfr and src_mfr_id <> @lSrcMfr and prod_line_id =
@lProdLine and accept_ind = '1' )

    -- get list of parts with access to org, the memb prod ref table needs to
be updated for these parts
    insert into #temp2 select distinct org_id , part_id from org_avail_part
    where part_id in (select distinct part_id from mfr_accept_part
    where mfr_id = @lPipelineMfr and src_mfr_id = @lSrcMfr and prod_line_id =
@lProdLine)
    and part_id not in (select distinct part_id from mfr_accept_part
    where mfr_id = @lPipelineMfr and src_mfr_id <> @lSrcMfr and prod_line_id =
@lProdLine and accept_ind = '1' )

    -- get a list of pipeline orgs belonging to this mfr which are affected by
this. these orgs need to be refreshed by recursive calling
    insert into #temp3 select distinct opr.org_id , opr.part_id from
org_prod_ref opr , org o
    where opr.org_id = o.org_id and o.pl_ind = '1'
    and opr.part_id in (select distinct part_id from mfr_accept_part
    where mfr_id = @lPipelineMfr and src_mfr_id = @lSrcMfr and prod_line_id =
@lProdLine)
    and opr.part_id not in (select distinct part_id from mfr_accept_part
    where mfr_id = @lPipelineMfr and src_mfr_id <> @lSrcMfr and prod_line_id =
@lProdLine and accept_ind = '1' )
*/
end

-- if commodity is being added, then insert all parts into the mfr_accept_part table
except for the parts which are already in there.
-- for these parts, turn avail_ind = 1

if @lCommodity > 0
begin
    -- update already existing parts
    update mfr_accept_part set avail_ind = '1'

    where mfr_id = @lPipelineMfr and src_mfr_id = @lSrcMfr and commodity_id =
@lCommodity

```

sp_pipeline_org_product_add.sql

```

if @lNew = 1
begin
    -- insert new parts for this prod line
    insert into mfr_accept_part ( mfr_id, part_id, src_mfr_id,
prod_line_id,
    commodity_id, accept_ind, dflt_src_ind, avail_ind, create_dtm,
last_maint_dtm )
    select @lPipelineMfr, part_id, @lSrcMfr , prod_line_id,
commodity_id, '0'
    , '0' , '1' , getdate(), getdate()
    from org_prod_ref where org_id = @lPipelineOrg and commodity_id =
@lCommodity
    and part_id not in (select distinct part_id from mfr_accept_part
    where mfr_id = @lPipelineMfr and src_mfr_id = @lSrcMfr and
commodity_id = @lCommodity)
end
/*
    -- get list of all parts with access to dflt members, the memb prod ref
table needs to be updated for these parts
    insert into #temp1 select distinct part_id from dflt_avail_part
    where part_id in (select distinct part_id from mfr_accept_part
    where mfr_id = @lPipelineMfr and src_mfr_id = @lSrcMfr and commodity_id =
@lCommodity)
    and part_id not in (select distinct part_id from mfr_accept_part
    where mfr_id = @lPipelineMfr and src_mfr_id <> @lSrcMfr and commodity_id =
@lCommodity and accept_ind = '1' )

    -- get list of parts with access to org, the memb prod ref table needs to
be updated for these parts
    insert into #temp2 select distinct org_id , part_id from org_avail_part
    where part_id in (select distinct part_id from mfr_accept_part
    where mfr_id = @lPipelineMfr and src_mfr_id = @lSrcMfr and commodity_id =
@lCommodity)
    and part_id not in (select distinct part_id from mfr_accept_part
    where mfr_id = @lPipelineMfr and src_mfr_id <> @lSrcMfr and commodity_id =
@lCommodity and accept_ind = '1' )

    -- get a list of pipeline orgs belonging to this mfr which are affected by
this. these orgs need to be refreshed by recursive calling
    insert into #temp3 select distinct opr.org_id , opr.part_id from
org_prod_ref opr , org o
    where opr.org_id = o.org_id and o.pl_ind = '1'
    and opr.part_id in (select distinct part_id from mfr_accept_part
    where mfr_id = @lPipelineMfr and src_mfr_id = @lSrcMfr and commodity_id =
@lCommodity)
    and opr.part_id not in (select distinct part_id from mfr_accept_part
    where mfr_id = @lPipelineMfr and src_mfr_id <> @lSrcMfr and commodity_id =
@lCommodity and accept_ind = '1' )
*/
end

-- if part is being added, then insert it into the mfr_accept_part table unless it
is already in there
-- for this part, turn avail_ind = 1

if @lPart > 0
begin
    -- update already existing part
    if exists ( select 1 from mfr_accept_part where mfr_id = @lPipelineMfr and
src_mfr_id = @lSrcMfr and part_id = @lPart)
    begin

```

```

        sp_pipeline_org_product_add.sql
        update mfr_accept_part set avail_ind = '1'
        where mfr_id = @lPipelineMfr and src_mfr_id = @lSrcMfr and part_id =
@lPart
        end
        else
        begin
            if @lNew = 1
            begin
                -- insert new parts for this prod line
                insert into mfr_accept_part ( mfr_id, part_id, src_mfr_id,
prod_line_id,
                commodity_id, accept_ind, dflt_src_ind, avail_ind,
create_dtm, last_maint_dtm )
                select @lPipelineMfr, @lPart , @lSrcMfr , prod_line_id,
commodity_id, '0' ,
                '0' , '1' , getdate(), getdate()
                from org_prod_ref where org_id = @lPipelineOrg and part_id =
@lPart
            end
        end
    /*
        -- get list of all parts with access to dflt members, the memb prod ref
table needs to be updated for these parts
        insert into #temp1 select distinct part_id from dflt_avail_part
        where part_id in (select distinct part_id from mfr_accept_part
        where mfr_id = @lPipelineMfr and src_mfr_id = @lSrcMfr and part_id = @lPart)
        and part_id not in (select distinct part_id from mfr_accept_part
        where mfr_id = @lPipelineMfr and src_mfr_id <> @lSrcMfr and part_id = @lPart
and accept_ind = '1' )

        -- get list of parts with access to org, the memb prod ref table needs to
be updated for these parts
        insert into #temp2 select distinct org_id , part_id from org_avail_part
        where part_id in (select distinct part_id from mfr_accept_part
        where mfr_id = @lPipelineMfr and src_mfr_id = @lSrcMfr and part_id = @lPart)
        and part_id not in (select distinct part_id from mfr_accept_part
        where mfr_id = @lPipelineMfr and src_mfr_id <> @lSrcMfr and part_id = @lPart
and accept_ind = '1' )

        -- get a list of pipeline orgs belonging to this mfr which are affected by
this. these orgs need to be refreshed by recursive calling
        insert into #temp3 select distinct opr.org_id , opr.part_id from
org_prod_ref opr , org o
        where opr.org_id = o.org_id and o.pl_ind = '1'
        and opr.part_id in (select distinct part_id from mfr_accept_part
        where mfr_id = @lPipelineMfr and src_mfr_id = @lSrcMfr and part_id = @lPart)
        and opr.part_id not in (select distinct part_id from mfr_accept_part
        where mfr_id = @lPipelineMfr and src_mfr_id <> @lSrcMfr and part_id = @lPart
and accept_ind = '1' )
    */
end
/*
-- add member prod ref entries for these parts
if (select count(*) from #temp1) > 0
begin
    declare memb_prod cursor for select part_id from #temp1
    open memb_prod
    fetch next from memb_prod into @lLocalPart
    while @@fetch_status = 0
    begin
        exec sp_dflt_avail_part_insert @lPipelineMfr , -1 , -1 , @lLocalPart
        fetch next from memb_prod into @lLocalPart
    end
end

```

```

                                sp_pipeline_org_product_add.sql
        close memb_prod
        deallocate memb_prod
end
-- add org prod rf entries for these parts
if (select count(*) from #temp2) > 0
begin
    declare org_prod cursor for select org_id , part_id from #temp2
    open org_prod
    fetch next from org_prod into @lLocalOrg , @lLocalPart
    while @@fetch_status = 0
    begin
        exec sp_org_avail_part_insert @lLocalOrg , -1 , -1 , @lLocalPart
        fetch next from org_prod into @lLocalOrg , @lLocalPart
    end
    close org_prod
    deallocate org_prod
end
--recursively run this procedure for all pipeline orgs dependent on this part
addition to pipelines
if (select count(*) from #temp3) > 0
begin
    declare pipe_org cursor for select org_id from #temp3
    open pipe_org
    fetch next from pipe_org into @lLocalOrg, @lLocalPart
    while @@fetch_status = 0
    begin
        exec sp_pipeline_org_product_add @lLocalOrg , -1 , -1 ,
@lLocalPart
    end
    close pipe_org
    deallocate pipe_org
end
*/

```

```

GO
SET QUOTED_IDENTIFIER OFF      SET ANSI_NULLS ON
GO
GRANT EXECUTE ON [dbo].[sp_pipeline_org_product_add] TO [public]
GO

```

```

SET QUOTED_IDENTIFIER OFF SET ANSI_NULLS ON
GO

```

```

if exists (select * from sysobjects where id =
object_id(N'[dbo].[sp_pipeline_org_product_remove]') and OBJECTPROPERTY(id,
N'IsProcedure') = 1)
drop procedure [dbo].[sp_pipeline_org_product_remove]
GO

```

```

/*
* sp_pipeline_org_product_remove
* this procedure will unsubscribe a pipeline org from a pipeline mfr
* it will recursively call itself to remove all depedent pipelines also
* uday jan 31 2000
* uday 8/12/00 modified part selection
* uday 8/25/00 fixed SP to get correct parts
*/

```

```

CREATE PROCEDURE sp_pipeline_org_product_remove ( @lPipelineOrg int, @lProdline int
= -1, @lCommodity int = -1 , @lPart int = -1 ) as

```

```

declare @lSrcMfr integer
declare @lPipelineMfr integer
declare @lLocalOrg integer
declare @lLocalPart integer

```

```

-- get mfr IDs for src and pipeline
select @lSrcMfr = mfr_id from org where org_id = @lPipelineOrg
select @lPipelineMfr = pl_mfr_id from org where org_id = @lPipelineOrg

```

```

-- check for pipeline mfr
if ( @lPipelineMfr is null or @lPipelineMfr <= 0 )
begin
    return
end

```

```

--check for error
if @lProdLine = -1 and @lCommodity = -1 and @lPart = -1
begin
    return
--
    raiserror ('The product ID supplied is not valid.' , 1 , 1)
end

```

```

create table #temp1 ( part_id int primary key)
create table #temp2 ( org_id int not null , part_id int not null )
--alter table #temp2 add constraint pk_temp2 primary key (org_id, part_id)

```

```

-- if prod line > 0 then get all prod line parts
if @lProdLine > 0
begin

```

```

    -- select all the parts in mfr accept part table with this src mfr as the
only source. this is needed for cascading the pipeline deletes
    insert into #temp1 select distinct part_id from mfr_accept_part
    where mfr_id = @lPipelineMfr and src_mfr_id = @lSrcMfr
    and part_id not in ( select distinct part_id from mfr_accept_part
    where mfr_id = @lPipelineMfr and src_mfr_id <> @lSrcMfr and accept_ind = '1'
)
--
    and part_id not in ( select distinct part_id from mfr_accept_part
--
    where mfr_id = @lSrcMfr and accept_ind = '1')
    and prod_line_id = @lProdline

```

```

                                sp_pipeline_org_product_remove.sql
-- get all pipeline orgs where these parts are pipelines to
insert into #temp2 select distinct org_prod_ref.org_id, part_id from
org_prod_ref, org
where org.org_id = org_prod_ref.org_id
and org.pl_ind = '1' and part_id in (select part_id from #temp1)
and prod_line_id = @lProdLine

end

-- if commodity > 0 then get all commodity parts
if @lCommodity > 0
begin
    -- select all the parts in mfr accept part table with this src mfr as the
only source. this is needed for cascading the pipeline deletes
insert into #temp1 select distinct part_id from mfr_accept_part
where mfr_id = @lPipelineMfr and src_mfr_id = @lSrcMfr
and part_id not in ( select distinct part_id from mfr_accept_part
where mfr_id = @lPipelineMfr and src_mfr_id <> @lSrcMfr and accept_ind = '1'
)
-- and part_id not in ( select distinct part_id from mfr_accept_part
-- where mfr_id = @lSrcMfr and accept_ind = '1')
and commodity_id = @lCommodity
select * from #temp1
-- get all pipeline orgs where these parts are pipelines to
insert into #temp2 select distinct org_prod_ref.org_id, part_id from
org_prod_ref, org
where org.org_id = org_prod_ref.org_id
and org.pl_ind = '1' and part_id in (select part_id from #temp1)
and commodity_id = @lCommodity

end

-- if part > 0 then get part details
if @lPart > 0
begin
    -- select all the parts in mfr accept part table with this src mfr as the
only source. this is needed for cascading the pipeline deletes
insert into #temp1 select distinct part_id from mfr_accept_part
where mfr_id = @lPipelineMfr and src_mfr_id = @lSrcMfr
and part_id not in ( select distinct part_id from mfr_accept_part
where mfr_id = @lPipelineMfr and src_mfr_id <> @lSrcMfr and accept_ind =
'1' )
-- and part_id not in ( select distinct part_id from mfr_accept_part
-- where mfr_id = @lSrcMfr and accept_ind = '1')
and part_id = @lPart

    -- get all pipeline orgs where these parts are pipelines to
insert into #temp2 select distinct org_prod_ref.org_id, part_id from
org_prod_ref, org
where org.org_id = org_prod_ref.org_id
and org.pl_ind = '1' and part_id in (select part_id from #temp1)
and part_id = @lPart

end

-- set all parts for this pl mfr ID avail ind = 0 and delete all rows from mfr
accept prod ref
-- turn avail ind = 0
update mfr_accept_part set avail_ind = '0'
where part_id in (select part_id from #temp1) and mfr_id = @lPipelineMfr and
src_mfr_id = @lSrcMfr

```



```

        sp_pipeline_org_product_remove.sql

        -- delete rows from dflt avail part and member prod ref tables
        -- if this src mfr is the only source for these parts
        delete from memb_prod_ref where part_id in (select part_id from #temp1) and
mfr_id = @lPipelineMfr

        --delete rows from org avail part and org prod ref tables
        -- these parts have only one source as already determined and the source is
terminating the pipeline
        delete from org_prod_ref where part_id in (select part_id from #temp1)

        -- now loop thru each of the orgs which are pipelines as selected earlier
into #temp2
        -- and recursively call this procedure to undo all the pipeline orgs

        declare pipe_org cursor for select org_id , part_id from #temp2
        open pipe_org
        fetch next from pipe_org into @lLocalOrg , @lLocalPart

        while @@fetch_status = 0
        begin
            exec sp_pipeline_org_product_remove @lLocalOrg , -1 , -1 ,
@lLocalPart
            fetch next from pipe_org into @lLocalOrg, @lLocalpart
        end

        close pipe_org
        deallocate pipe_org

GO

SET QUOTED_IDENTIFIER OFF      SET ANSI_NULLS ON
GO

GRANT EXECUTE ON [dbo].[sp_pipeline_org_product_remove] TO [public]
GO

```

```

SET QUOTED_IDENTIFIER OFF      sp_pipeline_org_remove.sql
GO                               SET ANSI_NULLS ON

if exists (select * from sysobjects where id =
object_id(N'[dbo].[sp_pipeline_org_remove]') and OBJECTPROPERTY(id, N'IsProcedure')
= 1)
drop procedure [dbo].[sp_pipeline_org_remove]
GO

/*
* sp_pipeline_org_remove
* this procedure will unsubscribe a pipeline org from a pipeline mfr
* it will recursively call itself to remove all deperdent pipelines also
* uday jan 31 2000
*/

CREATE PROCEDURE sp_pipeline_org_remove ( @lPipelineOrg int, @lDel int = 1 ) as

declare @lSrcMfr integer
declare @lPipelineMfr integer
declare @lOrg integer

-- get mfr IDs for src and pipeline
select @lSrcMfr = mfr_id from org where org_id = @lPipelineOrg
select @lPipelineMfr = pl_mfr_id from org where org_id = @lPipelineOrg

-- check for pipeline mfr
if ( @lPipelineMfr is null or @lPipelineMfr <= 0 )
begin
    return
end

-- select all the parts in mfr accept part table with this src mfr as the only
source. this is needed for cascading the pipeline deletes
select distinct part_id into #temp1 from mfr_accept_part
where mfr_id = @lPipelineMfr and src_mfr_id = @lSrcMfr
and part_id not in ( select distinct part_id from mfr_accept_part
where mfr_id = @lPipelineMfr and src_mfr_id <> @lSrcMfr and accept_ind = '1' )

-- get all pipeline orgs where these parts are pipelines to
select distinct org_prod_ref.org_id, part_id into #temp2 from org_prod_ref, org
where org.org_id = org_prod_ref.org_id
and org.pl_ind = '1' and part_id in (select part_id from #temp1)

-- set all parts for this pl mfr ID avail ind = 0 and delete all rows from mfr
accept prod ref
-- if @Del = 1 , then delete rows from mfr_accept_part - this is true unsubscribe,
else, turn avail ind = 0
if @lDel = 1
begin
    update mfr_accept_part set avail_ind = '0' where mfr_id =
@lPipelineMfr and src_mfr_id = @lSrcMfr
end
else
begin
    delete from mfr_accept_part where mfr_id = @lPipelineMfr and
src_mfr_id = @lSrcMfr
end
delete from mfr_accept_prod_ref where mfr_id = @lPipelineMfr and src_mfr_id =
@lSrcMfr

-- delete rows from dflt avail part and member prod ref tables
-- if this src mfr is the only source for these parts

```

```

                                sp_pipeline_org_remove.sql
if @lDel = 1
    begin
        delete from dflt_avail_part where part_id in (select part_id from
#temp1) and mfr_id = @lPipelineMfr
        end
        delete from memb_prod_ref where part_id in (select part_id from #temp1) and mfr_id =
@lPipelineMfr
--delete rows from org avail part and org prod ref tables
-- these parts have only one source as already determined and the source is
terminating the pipeline
if @lDel = 1
    begin
        delete from org_avail_part where part_id in (select part_id from
#temp1)
        end
        delete from org_prod_ref where part_id in (select part_id from #temp1)
-- now loop thru each of the orgs which are pipelines as selected earlier into
#temp2
-- and recursively call this procedure to undo all the pipeline orgs
declare pipe_org cursor for select org_id from #temp2
open pipe_org
fetch next from pipe_org into @lOrg
while @@fetch_status = 0
    begin
        exec sp_pipeline_org_remove @lOrg, 0
        fetch next from pipe_org into @lOrg
    end
close pipe_org
deallocate pipe_org

GO
SET QUOTED_IDENTIFIER OFF      SET ANSI_NULLS ON
GO

GRANT EXECUTE ON [dbo].[sp_pipeline_org_remove] TO [public]
GO

```

update_avail_tbls.sql

```
if exists (select * from sysobjects
           where name = "sp_mfr_accept_part_update_test"
           and type = "p")
  drop procedure sp_mfr_accept_part_update_test
go

create procedure sp_mfr_accept_part_update_test ( @mfr_id int, @src_mfr_id int ,
@add_flag varchar,

          @prod_ind varchar(2), @prod_string varchar(2000) )
as

--uday 8/25/00 mfr accept part update accepting and un accepting parts - batch
process to handle all mfrs affected in one batch
-- also does member, org availability
-- and tri state codes and pricing for all parts affected
declare @int_continue integer
declare @int_level integer
declare @iCounter integer
declare @SQLstring nvarchar(2000)

--drop table #pl_mfr_temp

select @int_continue = 1
select @int_level = 0
if @add_flag <> '1' select @add_flag = '0'

/**** create temp table *****/
CREATE TABLE #pl_mfr_temp(
    temp_id int identity(1,1),
    mfr_id int NOT NULL ,
    src_mfr_id int NOT NULL,
    org_id int NOT NULL,
    pl_level int
)
insert into #pl_mfr_temp values ( @mfr_id , @src_mfr_id , '0', @int_level)

--drop table #mfr_parts
CREATE TABLE #mfr_parts(
    part_id int NOT NULL ,
    prod_line_id int NOT NULL,
    commodity_id int,
    add_flag int
)

select @SQLstring = " insert into #mfr_parts "
select @SQLstring = @SQLstring + " select distinct
part_id,prod_line_id,commodity_id, " + @add_flag
select @SQLstring = @SQLstring + " from mfr_accept_part "
select @SQLstring = @SQLstring + " where mfr_id = " + cast(@mfr_id as nvarchar)
select @SQLstring = @SQLstring + " and src_mfr_id = " + cast(@src_mfr_id as
nvarchar)
--select @SQLstring = @SQLstring + " and part_id not in ( "
--select @SQLstring = @SQLstring + " select part_id from mfr_accept_part "
--select @SQLstring = @SQLstring + " where mfr_id = " + cast(@mfr_id as nvarchar)
--select @SQLstring = @SQLstring + " and src_mfr_id <> " + cast(@src_mfr_id as
nvarchar)
--select @SQLstring = @SQLstring + " and accept_ind = '1' and avail_ind = '1') "
if @prod_ind = 'P'
begin
select @SQLstring = @SQLstring + " and part_id in ( " + @prod_string + " ) "
end
end
```

update_avail_tbls.sql

```

if @prod_ind = 'C'
begin
select @SQLstring = @SQLstring + " and commodity_id in ( " + @prod_string + " ) "
end
if @prod_ind = 'L'
begin
select @SQLstring = @SQLstring + " and prod_line_id in ( " + @prod_string + " ) "
end
execute sp_executesql @SQLstring

--drop table #mfr_parts_add
CREATE TABLE #mfr_parts_add(
    mfr_id int NOT NULL,
    src_mfr_id int NOT NULL,
    part_id int NOT NULL ,
    prod_line_id int NOT NULL,
    commodity_id int,
    another_src int default 0
)

while @int_continue = 1
begin
select @int_level = @int_level + 1
begin
insert into #pl_mfr_temp select distinct a.pl_mfr_id,a.mfr_id,a.org_id,@int_level
from org a,
    #pl_mfr_temp b
where a.mfr_id= b.mfr_id
    and a.pl_ind = 1
    and a.pl_mfr_id is not null
    and @int_level-1= b.pl_level
    and not exists(select 'x' from #pl_mfr_temp c
        where a.pl_mfr_id = c.mfr_id
        and a.mfr_id = c.src_mfr_id)

select @int_continue = count(*)
from org c,
    #pl_mfr_temp d
where c.mfr_id= d.mfr_id
    and c.pl_ind = 1
    and @int_level-1= d.pl_level
end

if @int_continue > 0
begin
select @int_continue = 1
end
else
begin
select @int_continue = 0
end
end

insert into #mfr_parts_add
select
    a.mfr_id,
    a.src_mfr_id,
    b.part_id,
    b.prod_line_id,
    b.commodity_id,
    0
from #pl_mfr_temp a,
    #mfr_parts b,

```

update_avail_tbls.sql

```

org_avail_part c
where a.org_id = c.org_id
and b.part_id = c.part_id
and a.pl_level > 0

```

```

--add 0 level parts so that all processing can be done together
insert into #mfr_parts_add select @mfr_id, @src_mfr_id, part_id, prod_line_id,
commodity_id , 0 from #mfr_parts

```

```

--update another_src field based on whether the part has another valid src other
than the one we are checking for

```

```

update #mfr_parts_add set another_src = 1
from mfr_accept_part p
where #mfr_parts_add.mfr_id = p.mfr_id
and #mfr_parts_add.src_mfr_id <> p.src_mfr_id
and #mfr_parts_add.part_id = p.part_id
and p.accept_ind = '1'
and p.avail_ind = '1'

```

```

update mfr_accept_part set accept_ind = @add_flag
from #pl_mfr_temp t , #mfr_parts_add b
where b.mfr_id = mfr_accept_part.mfr_id
and b.part_id = mfr_accept_part.part_id
and b.prod_line_id = mfr_accept_part.prod_line_id
and mfr_accept_part.src_mfr_id = t.src_mfr_id
and t.mfr_id = b.mfr_id
and t.pl_level = 0

```

```

update mfr_accept_part set avail_ind = @add_flag
from #pl_mfr_temp t, #mfr_parts_add b
where b.mfr_id = mfr_accept_part.mfr_id
and b.part_id = mfr_accept_part.part_id
and b.prod_line_id = mfr_accept_part.prod_line_id
and b.src_mfr_id = mfr_accept_part.src_mfr_id
and t.mfr_id = b.mfr_id
and t.src_mfr_id = b.src_mfr_id
and t.pl_level > 0

```

```

-- set up ref table for mfr accept part table

```

```

if @add_flag = '1'
begin
insert mfr_accept_prod_ref
(mfr_id, src_mfr_id, prod_line_id, commodity_id, part_id,
mfr_ts_cd, prod_line_ts_cd, comm_ts_cd,
accept_ind, dflt_src_ind, avail_ind, src_price)
select distinct @mfr_id, @src_mfr_id, prod_line_id, commodity_id, part_id,
0, 0, 0,
1, 0, 1, 0
from #mfr_parts
end

```

```

else
begin
delete from mfr_accept_prod_ref
where part_id in (select part_id from #mfr_parts)
and mfr_id = @mfr_id
and src_mfr_id = @src_mfr_id
end

```

```

--set up ref tables for all other levels

```

```

update mfr_accept_part set avail_ind = @add_flag
from #pl_mfr_temp p , #mfr_parts_add m
where p.mfr_id = m.mfr_id

```

```

                                update_avail_tbls.sql
and p.src_mfr_id = m.src_mfr_id
and mfr_accept_part.mfr_id = m.mfr_id
and mfr_accept_part.src_mfr_id = m.src_mfr_id
and mfr_accept_part.part_id = m.part_id
and p.pl_level > 0

if @add_flag = '1'
begin
    insert mfr_accept_prod_ref
    ( mfr_id, src_mfr_id, prod_line_id, commodity_id, part_id,
      mfr_ts_cd, prod_line_ts_cd, comm_ts_cd,
      accept_ind, dflt_src_ind, avail_ind, src_price)
    select m.mfr_id, m.src_mfr_id, m.prod_line_id, m.commodity_id, m.part_id,
    0, 0, 0,
    a.accept_ind , 0, 1, 0
    from #mfr_parts_add m , #pl_mfr_temp p, mfr_accept_part a
    where p.mfr_id = m.mfr_id
    and p.src_mfr_id = m.src_mfr_id
    and m.mfr_id = a.mfr_id
    and m.src_mfr_id = a.src_mfr_id
    and m.part_id = a.part_id
    and p.pl_level > 0
end
else
begin
    delete mfr_accept_prod_ref from mfr_accept_prod_ref
    inner join #mfr_parts_add on (#mfr_parts_add.mfr_id =
mfr_accept_prod_ref.mfr_id
                                and
#mfr_parts_add.src_mfr_id = mfr_accept_prod_ref.src_mfr_id
                                and
#mfr_parts_add.part_id = mfr_accept_prod_ref.part_id)
    inner join #pl_mfr_temp on (#pl_mfr_temp.mfr_id = mfr_accept_prod_ref.mfr_id
                                and
#pl_mfr_temp.src_mfr_id = mfr_accept_prod_ref.src_mfr_id)
    where #pl_mfr_temp.pl_level > 0
end

-- set up ref table for memb part table for 0 level mfr
delete memb_prod_ref from memb_prod_ref
    inner join #mfr_parts_add on #mfr_parts_add.mfr_id = memb_prod_ref.mfr_id
    and #mfr_parts_add.part_id = memb_prod_ref.part_id
    and #mfr_parts_add.commodity_id = memb_prod_ref.commodity_id
    and #mfr_parts_add.prod_line_id = memb_prod_ref.prod_line_id
    and #mfr_parts_add.another_src = 0

if @add_flag = '1'
begin
    --level > 0
    insert memb_prod_ref ( mfr_id, part_id, prod_line_id, commodity_id,
    prod_line_ts_cd, comm_ts_cd ,
                                list_price, sale_price, final_price, disc_ind)
    select distinct m.mfr_id, m.part_id, m.prod_line_id, m.commodity_id, 0, 0,
    0, 0, 0, '0'
    from #mfr_parts_add m , dflt_avail_part d
    where m.mfr_id = d.mfr_id
    and m.part_id = d.part_id
    and m.commodity_id = d.commodity_id
    and m.prod_line_id = d.prod_line_id
    and m.another_src = 0
end

--set up org prod ref tables

```

```

                                update_avail_tbls.sql
delete org_prod_ref from org_prod_ref
    inner join org on org.org_id = org_prod_ref.org_id
    inner join #mfr_parts_add on #mfr_parts_add.part_id = org_prod_ref.part_id
    and #mfr_parts_add.commodity_id = org_prod_ref.commodity_id
    and #mfr_parts_add.prod_line_id = org_prod_ref.prod_line_id
    where org.mfr_id = #mfr_parts_add.mfr_id
    and #mfr_parts_add.another_src = 0
if @add_flag = '1'
begin
insert org_prod_ref ( org_id, part_id, prod_line_id, commodity_id, prod_line_ts_cd,
comm_ts_cd ,
                        list_price, sale_price, final_price, disc_ind )
select distinct op.org_id, m.part_id, m.prod_line_id, m.commodity_id, 0, 0,
                        0, 0, 0, '0'
from #mfr_parts_add m , org_avail_part op, org o
where m.mfr_id = o.mfr_id
and op.org_id = o.org_id
and op.part_id = m.part_id
and op.commodity_id = m.commodity_id
and op.prod_line_id = m.prod_line_id
and m.another_src = 0
end

-- set up tri states for mfr, memb and org table
select @iCounter = 1
while @iCounter <= (select count(*) from #pl_mfr_temp)
begin
    select @mfr_id = mfr_id, @src_mfr_id = src_mfr_id
    from #pl_mfr_temp
    where temp_id = @iCounter
    select @iCounter = @iCounter + 1
    exec sp_mfr_accept_tri_state_test @mfr_id, @src_mfr_id
    exec sp_dflt_tri_state_test @mfr_id
    exec sp_org_tri_state_test @mfr_id
end

-- run price calcs for all the parts in the mfr_parts_add table
if @add_flag = '1'
begin
    exec sp_mfr_price_calc_test
    exec sp_mfr_dflt_price_calc_test
end

go
grant execute on sp_mfr_accept_part_update_test to public
go

```


update_org_avail.sql

```
if exists (select * from sysobjects
           where name = "sp_org_avail_part_test"
           and type = "p")
    drop procedure sp_org_avail_part_test
go

create procedure sp_org_avail_part_test ( @mfr_id int, @org_id int , @add_flag
varchar,

           @prod_ind varchar(2), @prod_string varchar(2000) )
as

--uday 9/6/00 org avail parts adding and deleting parts - batch process to handle
all mfrs affected in one batch
-- also does member, org availability
-- and tri state codes and pricing for all parts affected
declare @int_continue integer
declare @int_level integer
declare @iCounter integer
declare @SQLstring nvarchar(2000)
declare @src_mfr_id integer

select @int_continue = 1
select @int_level = 0
if @add_flag <> '1' select @add_flag = '0'
--set src mfr as 0 - this does not matter since no src related stuff for the
original org is affected in any way
select @src_mfr_id = 0

/**** create temp table *****/
CREATE TABLE #pl_mfr_temp(
    temp_id int identity(1,1),
    mfr_id int NOT NULL ,
    src_mfr_id int NOT NULL,
    org_id int NOT NULL,
    pl_level int)

insert into #pl_mfr_temp values ( @mfr_id , @src_mfr_id , 0 , @int_level)

--drop table #mfr_parts
CREATE TABLE #mfr_parts(
    part_id int NOT NULL ,
    prod_line_id int NOT NULL,
    commodity_id int,
    add_flag int
)

select @SQLstring = " insert into #mfr_parts "
select @SQLstring = @SQLstring + "select distinct
part_id,prod_line_id,commodity_id, " + @add_flag
select @SQLstring = @SQLstring + " from mfr_accept_part "
select @SQLstring = @SQLstring + " where mfr_id = " + cast(@mfr_id as nvarchar)
select @SQLstring = @SQLstring + " and src_mfr_id = " + cast(@src_mfr_id as
nvarchar)
--select @SQLstring = @SQLstring + " and part_id not in ( "
--select @SQLstring = @SQLstring + " select part_id from mfr_accept_part "
--select @SQLstring = @SQLstring + " where mfr_id = " + cast(@mfr_id as nvarchar)
--select @SQLstring = @SQLstring + " and src_mfr_id <> " + cast(@src_mfr_id as
nvarchar)
--select @SQLstring = @SQLstring + " and accept_ind = '1' and avail_ind = '1') "
if @prod_ind = 'p'
begin
```

```

update_org_avail.sql
select @SQLstring = @SQLstring + " and part_id in ( " + @prod_string + " ) "
end
if @prod_ind = 'C'
begin
select @SQLstring = @SQLstring + " and commodity_id in ( " + @prod_string + " ) "
end
if @prod_ind = 'L'
begin
select @SQLstring = @SQLstring + " and prod_line_id in ( " + @prod_string + " ) "
end
execute sp_executesql @SQLstring

--set up ref table for 0 level for org only, the deletes are after mfr_parts_add
table is populated
if @add_flag = '1'
begin
insert into org_avail_part
select @org_id, part_id, commodity_id, prod_line_id , getdate()
from #mfr_parts
insert into org_prod_ref
select @org_id, part_id, prod_line_id, commodity_id, 0, 0, 0, 0, 0, 0, 0, 0,
null, null, null, null, null, null
from #mfr_parts
end

--drop table #mfr_parts_add
CREATE TABLE #mfr_parts_add(
    mfr_id int NOT NULL,
    src_mfr_id int NOT NULL,
    part_id int NOT NULL ,
    prod_line_id int NOT NULL,
    commodity_id int,
    another_src int default 0
)

while @int_continue = 1
begin
select @int_level = @int_level + 1
begin
insert into #pl_mfr_temp select distinct a.pl_mfr_id,a.mfr_id,a.org_id,@int_level
from org a,
    #pl_mfr_temp b
where a.mfr_id= b.mfr_id
    and a.pl_ind = 1
    and a.pl_mfr_id is not null
    and @int_level-1= b.pl_level
    and not exists(select 'x' from #pl_mfr_temp c
        where a.pl_mfr_id = c.mfr_id
        and a.mfr_id = c.src_mfr_id)

select @int_continue = count(*)
from org c,
    #pl_mfr_temp d
where c.mfr_id= d.mfr_id
    and c.pl_ind = 1
    and @int_level-1= d.pl_level
-- for orgs, if level = 1 then only 1 org supplied to this SP has to be in
the list
if @int_level = 1
begin
delete from #pl_mfr_temp where org_id <> @org_id and pl_level = 1
end
end
end

```

update_org_avail.sql

```

if @int_continue > 0
begin
    select @int_continue = 1
end
else
begin
    select @int_continue = 0
end
end

insert into #mfr_parts_add
select
    a.mfr_id,
    a.src_mfr_id,
    b.part_id,
    b.prod_line_id,
    b.commodity_id,
    0
from #pl_mfr_temp a,
     #mfr_parts b,
     org_avail_part c
where a.org_id = c.org_id
     and b.part_id = c.part_id
     and a.pl_level > 0

--add 0 level parts so that all processing can be done together
--insert into #mfr_parts_add select @mfr_id, @src_mfr_id, part_id, prod_line_id,
commodity_id , 0 from #mfr_parts

--update another_src field based on whether the part has another valid src other
than the one we are checking for
update #mfr_parts_add set another_src = 1
    from mfr_accept_part p
    where #mfr_parts_add.mfr_id = p.mfr_id
    and #mfr_parts_add.src_mfr_id <> p.src_mfr_id
    and #mfr_parts_add.part_id = p.part_id
    and p.accept_ind = '1'
    and p.avail_ind = '1'

if @add_flag = '0'
begin
    delete org_prod_ref from org_prod_ref
    inner join #mfr_parts on #mfr_parts.part_id = org_prod_ref.part_id
    and #mfr_parts.commodity_id = org_prod_ref.commodity_id
    and #mfr_parts.prod_line_id = org_prod_ref.prod_line_id
    and org_prod_ref.org_id = @org_id
    delete org_avail_part from org_avail_part
    inner join #mfr_parts on #mfr_parts.part_id = org_avail_part.part_id
    and #mfr_parts.commodity_id = org_avail_part.commodity_id
    and #mfr_parts.prod_line_id = org_avail_part.prod_line_id
    and org_avail_part.org_id = @org_id
end

/* - no accept table update for org details
update mfr_accept_part set accept_ind = @add_flag
    from #pl_mfr_temp t , #mfr_parts_add b
    where b.mfr_id = mfr_accept_part.mfr_id
    and b.part_id = mfr_accept_part.part_id
    and b.prod_line_id = mfr_accept_part.prod_line_id
    and mfr_accept_part.src_mfr_id = t.src_mfr_id
    and t.mfr_id = b.mfr_id
    and t.pl_level = 0

```

update_org_avail.sql

```

update mfr_accept_part set avail_ind = @add_flag
    from #pl_mfr_temp t, #mfr_parts_add b
    where b.mfr_id = mfr_accept_part.mfr_id
    and b.part_id = mfr_accept_part.part_id
    and b.prod_line_id = mfr_accept_part.prod_line_id
    and b.src_mfr_id = mfr_accept_part.src_mfr_id
    and t.mfr_id = b.mfr_id
    and t.src_mfr_id = b.src_mfr_id
    and t.pl_level > 0

-- set up ref table for mfr accept part table
if @add_flag = '1'
begin
insert mfr_accept_prod_ref
    ( mfr_id, src_mfr_id, prod_line_id, commodity_id, part_id,
      mfr_ts_cd, prod_line_ts_cd, comm_ts_cd,
      accept_ind, dflt_src_ind, avail_ind, src_price)
select distinct @mfr_id, @src_mfr_id, prod_line_id, commodity_id, part_id,
    0, 0, 0,
    1, 0, 1, 0
from #mfr_parts
end
else
begin
delete from mfr_accept_prod_ref
    where part_id in (select part_id from #mfr_parts)
    and mfr_id = @mfr_id
    and src_mfr_id = @src_mfr_id
end
*/

--set up ref tables for all other levels
update mfr_accept_part set avail_ind = @add_flag
    from #pl_mfr_temp p, #mfr_parts_add m
    where p.mfr_id = m.mfr_id
    and p.src_mfr_id = m.src_mfr_id
    and mfr_accept_part.mfr_id = m.mfr_id
    and mfr_accept_part.src_mfr_id = m.src_mfr_id
    and mfr_accept_part.part_id = m.part_id
    and p.pl_level > 0
if @add_flag = '1'
begin
insert mfr_accept_prod_ref
    ( mfr_id, src_mfr_id, prod_line_id, commodity_id, part_id,
      mfr_ts_cd, prod_line_ts_cd, comm_ts_cd,
      accept_ind, dflt_src_ind, avail_ind, src_price)
select m.mfr_id, m.src_mfr_id, m.prod_line_id, m.commodity_id, m.part_id,
    0, 0, 0,
    a.accept_ind, 0, 1, 0
from #mfr_parts_add m, #pl_mfr_temp p, mfr_accept_part a
    where p.mfr_id = m.mfr_id
    and p.src_mfr_id = m.src_mfr_id
    and m.mfr_id = a.mfr_id
    and m.src_mfr_id = a.src_mfr_id
    and m.part_id = a.part_id
    and p.pl_level > 0
end
else
begin
delete mfr_accept_prod_ref from mfr_accept_prod_ref
inner join #mfr_parts_add on (#mfr_parts_add.mfr_id =

```

```

update_org_avail.sql

mfr_accept_prod_ref.mfr_id
#mfr_parts_add.src_mfr_id = mfr_accept_prod_ref.src_mfr_id
#mfr_parts_add.part_id = mfr_accept_prod_ref.part_id)
inner join #pl_mfr_temp on (#pl_mfr_temp.mfr_id = mfr_accept_prod_ref.mfr_id
#pl_mfr_temp.src_mfr_id = mfr_accept_prod_ref.src_mfr_id)
where #pl_mfr_temp.pl_level > 0
end

-- set up ref table for memb part table for 0 level mfr
delete memb_prod_ref from memb_prod_ref
inner join #mfr_parts_add on #mfr_parts_add.mfr_id = memb_prod_ref.mfr_id
and #mfr_parts_add.part_id = memb_prod_ref.part_id
and #mfr_parts_add.commodity_id = memb_prod_ref.commodity_id
and #mfr_parts_add.prod_line_id = memb_prod_ref.prod_line_id
and #mfr_parts_add.another_src = 0
if @add_flag = '1'
begin
--level > 0
insert memb_prod_ref ( mfr_id, part_id, prod_line_id, commodity_id,
prod_line_ts_cd, comm_ts_cd ,
list_price, sale_price, final_price, disc_ind)
select distinct m.mfr_id, m.part_id, m.prod_line_id, m.commodity_id, 0, 0,
0, 0, 0, '0'
from #mfr_parts_add m , dflt_avail_part d
where m.mfr_id = d.mfr_id
and m.part_id = d.part_id
and m.commodity_id = d.commodity_id
and m.prod_line_id = d.prod_line_id
and m.another_src = 0
end

--set up org prod ref tables
delete org_prod_ref from org_prod_ref
inner join org on org.org_id = org_prod_ref.org_id
inner join #mfr_parts_add on #mfr_parts_add.part_id = org_prod_ref.part_id
and #mfr_parts_add.commodity_id = org_prod_ref.commodity_id
and #mfr_parts_add.prod_line_id = org_prod_ref.prod_line_id
where org.mfr_id = #mfr_parts_add.mfr_id
and #mfr_parts_add.another_src = 0
if @add_flag = '1'
begin
insert org_prod_ref ( org_id, part_id, prod_line_id, commodity_id, prod_line_ts_cd,
comm_ts_cd ,
list_price, sale_price, final_price, disc_ind )
select distinct op.org_id, m.part_id, m.prod_line_id, m.commodity_id, 0, 0,
0, 0, 0, '0'
from #mfr_parts_add m , org_avail_part op, org o
where m.mfr_id = o.mfr_id
and op.org_id = o.org_id
and op.part_id = m.part_id
and op.commodity_id = m.commodity_id
and op.prod_line_id = m.prod_line_id
and m.another_src = 0
end

-- set up tri states for mfr, memb and org table
exec sp_org_tri_state_test @mfr_id
select @iCounter = 1
while @iCounter <= (select count(*) from #pl_mfr_temp)
begin

```

```

                                update_org_avail.sql
select @mfr_id = mfr_id, @src_mfr_id = src_mfr_id
from #pl_mfr_temp
where temp_id = @iCounter
select @iCounter = @iCounter + 1
exec sp_mfr_accept_tri_state_test @mfr_id, @src_mfr_id
exec sp_dflt_tri_state_test @mfr_id
exec sp_org_tri_state_test @mfr_id
end

-- run price calcs for all the parts in the mfr_parts_add table
if @add_flag = '1'
begin
    exec sp_mfr_price_calc_test
    exec sp_mfr_dflt_price_calc_test
end

go
grant execute on sp_org_avail_part_test to public
go

```

BOrg.cls

VERSION 1.0 CLASS

BEGIN

MultiUse = -1 'True
Persistable = 0 'NotPersistable
DataBindingBehavior = 0 'vbNone
DataSourceBehavior = 0 'vbNone
MTSTransactionMode = 2 'RequiresTransaction

END

Attribute VB_Name = "BOrg"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = True
Option Explicit
Option Base 0

=====

' this function adds an Organization for a mfr
' takes mfr ID and Org table columns as arguments
' returns a record set with Org details

=====

' business function controls transaction for adding an org

Public Function AddOrg(_
ByVal alManfID As Long, ByVal asOrgName As String, ByVal asPipelineInd As
String, _
ByVal alPipelineID As Long, ByVal asPipelineSubscriptionCd As String, _
ByVal asDescription As String, ByVal asAddr1 As String, ByVal asAddr2 As
String, _
ByVal asCity As String, ByVal asState As String, ByVal asPostalCode As
String, _
ByVal asCountry As String, ByVal asPhone As String, ByVal asFax As String,
ByVal asEmailAddr As String, _
ByVal aswebDiscountIndicator As String, ByVal asDefaultShipCostIndicator As
String, _
ByVal asAltImagesIndicator As String) _
As ADODB.Recordset

On Error GoTo AddOrgErr

Dim lOrgID As Long
Dim oSeq As DawnSeq.BSequence
Dim oOrg As DawnOrg.DBOrg
Dim sCheckOrg As String
Dim lRow As Long
Dim sSelect As String
Dim oRs As ADODB.Recordset

'check org name nulls

If IsEmpty(asOrgName) Or IsNull(asOrgName) Then
RaiseResError DawnOrgErrors.ecOrgNameNotValid, TypeName(Me), "AddOrg"
End If

'check to make sure Org name is unique

sSelect = "SELECT org_id FROM org WHERE org_nme='" & GetSQLStr(asOrgName) & "'"
sSelect = sSelect & " AND active_ind = '1' AND mfr_id = " & CStr(alManfID)
Set oRs = GetSQLRecordset(sSelect)
If Not oRs.EOF Then
RaiseResError DawnOrgErrors.ecOrgNameNotUnique, TypeName(Me), "AddOrg"
End If

'get the org ID from DawnSeq component

Set oSeq = CtxCreateObject("DawnSeq.BSequence")
lOrgID = oSeq.GetNextInSequence("org") 'no error check - dawnseq will raise
error is necessaary

BOrg.cls

```

Set oSeq = Nothing

'check the pipeline indicator and generate a subscription code if necessary.
If asPipelineInd = "1" Then
    asPipelineSubscriptionCd = Crypt("E", "THEROCK", CStr(lorgID)) &
CStr(lorgID)
End If

'start DB component and call addorg function
Set oOrg = CtxCreateObject("DawnOrg.DBOrg")
lRow = oOrg.AddOrg( _
    alManfID, lorgID, asOrgName, asPipelineInd, alPipelineID, _
    asPipelineSubscriptionCd, asDescription, asAddr1, asAddr2, asCity, _
    asState, asPostalCode, asCountry, asPhone, asFax, asEmailAddr,
asWebDiscountIndicator, _
    asDefaultShipCostIndicator, asAltImagesIndicator)

'check rows affected - ie if row was inserted
If lRow <> 1 Then
    RaiseResError DawnOrgErrors.ecAddOrgFailed, TypeName(Me), "AddOrg"
End If

'get an RS for the same data
Set AddOrg = oOrg.GetOrgByID(lorgID)

'end trans
CtxSetComplete

'cleanup
Set oOrg = Nothing
Set oSeq = Nothing
Set oRs = Nothing
Exit Function

'error handling
AddOrgErr:
CtxSetAbort
Set oOrg = Nothing
Set oSeq = Nothing
Set oRs = Nothing
RaiseError TypeName(Me), "AddOrg"

End Function

'=====
'This function adds an address to an Org
'INPUTS: all shipping columns
'RETURNS: Recordset = specific address details
'=====
Public Function AddShipToAddress(ByVal alOrgID As Long, ByVal asAddr1 As String,
ByVal asAddr2 As String, _
    ByVal asCity As String, ByVal alState As Long, ByVal asZip As
String, ByVal alCountry As Long) _
    As ADODB.Recordset
On Error GoTo ErrHandler

'var
Dim oDBShip As DawnOrg.DBShipToAddr
Dim oSeq As DawnSeq.BSequence
Dim lShipID As Long
Dim lRow As Long

'get sequence for ship ID

```



```

                                BOrg.cls
' Set oSeq = CtxCreateObject("DawnSeq.BSequence")
' lShipID = oSeq.GetNextInSequence("address")
' Set oSeq = Nothing
'
' build business component and call add function
' Set oDBShip = CtxCreateObject("DawnOrg.DBShipToAddr")
' lRow = oDBShip.AddShipToAddress(alOrgID, lShipID, asAddr1, asAddr2, _
                                asCity, alState, asZip,
alCountry)
'
' check to make sure the add worked - rows affected
' If lRow <> 1 Then
'     RaiseResError DawnOrgErrors.ecAddShipAddressFailed, TypeName(Me),
"AddShipToAddress"
' End If
'
' get RS and end trans
' Set AddShipToAddress = oDBShip.GetShipToAddress(alOrgID, lShipID)
' CtxSetComplete
' Set oDBShip = Nothing
' Set oSeq = Nothing
' Exit Function
'
' error
ErrHandler:
' CtxSetAbort
' Set oDBShip = Nothing
' Set oSeq = Nothing
' RaiseError TypeName(Me), "AddShipToAddress"

```

End Function

```

'=====
' This function updates an address to an Org
' INPUTS: all shipping columns
' RETURNS: Recordset = specific address details
'=====
Public Function UpdateShipToAddress( _
    ByVal alOrgID As Long, ByVal alShipID As Long, ByVal asAddr1 As String,
ByVal asAddr2 As String, _
    ByVal asCity As String, ByVal alState As Long, ByVal asZip As String, ByVal
alCountry As Long, _
    ByVal adMaintDate As Date) As ADODB.Recordset
On Error GoTo ErrHandler
'
' var
' Dim oDBShip As DawnOrg.DBShipToAddr
' Dim lRow As Long
'
' build business component and call add function
' Set oDBShip = CtxCreateObject("DawnOrg.DBShipToAddr")
' lRow = oDBShip.UpdateShipToAddress(alOrgID, alShipID, asAddr1, asAddr2, _
                                asCity, alState, asZip,
alCountry, adMaintDate)
'
' check to make sure the add worked - rows affected
' If lRow <> 1 Then
'     RaiseResError DawnOrgErrors.ecUpdateShipAddressFailed, TypeName(Me),
"UpdateShipToAddress"
' End If
'
' get RS and end trans

```

```

                                Borg.cls
'    Set UpdateShipToAddress = oDBShip.GetShipToAddress(alOrgID, alShipID)
'    CtxSetComplete
'    Set oDBShip = Nothing
Exit Function

'error
ErrorHandler:
    CtxSetAbort
'    Set oDBShip = Nothing
'    RaiseError TypeName(Me), "UpdateShipToAddress"

End Function

'=====
'This function deletes an address for an Org
'INPUTS: org ID , ship ID and maintenance date
'RETURNS: none
'=====
Public Sub DeleteShipToAddress(ByVal alOrgID As Long, ByVal alShipID As Long, _
                                ByVal adMaintDate As Date)
On Error GoTo ErrorHandler

'var
Dim oDBShip As DawnOrg.DBShipToAddr
Dim oDBOrg As DawnOrg.DBOrg
Dim lRow As Long
Dim sUpdate As String
Dim lNull As Long
Dim dMaintDate As Date
Dim oRS As ADODB.Recordset
Dim lPrefShipID As Long

'build business component and call add function
Set oDBShip = CtxCreateObject("DawnOrg.DBShipToAddr")
Set oDBOrg = CtxCreateObject("DawnOrg.DBOrg")
lRow = oDBShip.DeleteShipToAddress(alOrgID, alShipID, adMaintDate)

'check to make sure the add worked - rows affected
If lRow <> 1 Then
    RaiseResError DawnOrgErrors.ecDeleteShipAddressFailed, TypeName(Me),
"DeleteShipToAddress"
End If

'make sure this id is removed from the ORG table if this is the preferred
address
Set oRS = oDBOrg.GetOrgByID(alOrgID)
dMaintDate = oRS("last_maint_dtm")
lPrefShipID = oRS("pref_ship_to_addr_id")
If lPrefShipID = alShipID Then
    sUpdate = "UPDATE org SET pref_ship_to_addr_id=null" & _
               " , last_maint_dtm=" & GetSQLDate(Now) & _
               " WHERE org_id=" & CStr(alOrgID) & _
               " AND last_maint_dtm=" & GetSQLDate('dMaintDate)
    'update org and check for errors
    lRow = RunSQL(sUpdate)
    If lRow <> 1 Then
        RaiseResError DawnOrgErrors.ecShipToaddressNotRemoved,
TypeName(Me), "DeleteShipToAddress"
    End If
End If

'get RS and end trans
CtxSetComplete

```

Borg.cls

```
' Set oDBShip = Nothing
' Set oDBOrg = Nothing
' Set oRS = Nothing
Exit Sub
```

```
'error
ErrorHandler:
' CtxSetAbort
' Set oDBShip = Nothing
' Set oDBOrg = Nothing
' Set oRS = Nothing
RaiseError TypeName(Me), "DeleteShipToAddress"
```

End Sub

```
'=====
'***** THIS FUNCTION SHOULD BE DELETED. *****
'This function sets the default ship to address or an organization
'INPUTS: org ID , ship ID and maintenance date for the Org - not the ship address
'RETURNS: none
'=====
```

```
Public Sub SetDefaultShipToAddress(ByVal aOrgID As Long, ByVal aShipID As Long, _
ByVal adMaintDate As Date)
```

On Error GoTo ErrorHandler

```
'var
' Dim oDBOrg As DawnOrg.DBOrg
' Dim lRow As Long
```

```
'build business component and call add function
' Set oDBOrg = CtxCreateObject("DawnOrg.DBOrg")
' lRow = oDBOrg.SetDefaultShipToAddress(aOrgID, aShipID, adMaintDate)
```

```
'check to make sure the add worked - rows affected
' If lRow <> 1 Then
' RaiseResError DawnOrgErrors.ecSetDefaultAddressFailed, TypeName(Me),
"SetDefaultShipToAddress"
' End If
```

```
'get RS and end trans
CtxSetComplete
' Set oDBOrg = Nothing
Exit Sub
```

```
'error
ErrorHandler:
' CtxSetAbort
' Set oDBOrg = Nothing
RaiseError TypeName(Me), "SetDefaultShipToAddress"
```

End Sub

```
'=====
' this function updates an Organization for a mfr
' takes mfr ID and Org table columns as arguments
' returns a record set with Org details
'business function to update org, controls transaction
' 7/6/00 chrubeky - added call to sp_update_price function if the use_web_disc_ind
changes
'=====
```

```
Public Function UpdOrg(ByVal aOrgID As Long, ByVal asOrgName As String, _
ByVal asPipelineInd As String, ByVal alPipelineID As Long, _
```

```

                                BOrg.Cls
        ByVal asPipelineSubscriptionCd As String, ByVal asDescription As String,
-       ByVal asAddr1 As String, ByVal asAddr2 As String, _
        ByVal asCity As String, ByVal asState As String, ByVal asPostalCode As
String, -
        ByVal asCountry As String, ByVal asPhone As String, ByVal asFax As
String, -
        ByVal asEmailAddr As String, ByVal asWebDiscountIndicator As String, _
        ByVal asDefaultShipCostIndicator As String, ByVal asMaintDate As Date, _
        ByVal asAltImagesIndicator As String) As ADODB.Recordset

    On Error GoTo UpdOrgErr

    Dim oOrg As DawnOrg.DBORG
    Dim oRS As ADODB.Recordset
    Dim lRow As Long
    Dim sSelect As String
    Dim lManfID As Long
    Dim bCallSP As Boolean
    Dim lProdLineID As Long
    Dim sSQL As String

    'check org name nulls
    If IsEmpty(asOrgName) Or IsNull(asOrgName) Then
        RaiseResError DawnOrgErrors.ecOrgNameNotValid, TypeName(Me), "UpdOrg"
    End If

    'get the mfr_id associated with this org.
    sSelect = "SELECT mfr_id, use_web_disc_ind from org where org_id = " &
CStr(alOrgID)
    Set oRS = GetSQLRecordset(sSelect)
    If Not oRS.EOF Then
        lManfID = oRS("mfr_id")
        If oRS!use_web_disc_ind <> asWebDiscountIndicator Then
            bCallSP = True
        End If
    Else
        RaiseResError DawnOrgErrors.ecManfIDNotFound, TypeName(Me), "UpdOrg"
    End If

    'check to make sure Org name is unique
    sSelect = "SELECT org_id FROM org WHERE org_nme='" & GetSQLStr(asOrgName) & "'"
& _
        " AND org_id <> " & CStr(alOrgID) & " AND mfr_id = " &
CStr(lManfID)
    Set oRS = GetSQLRecordset(sSelect)
    If Not oRS.EOF Then
        RaiseResError DawnOrgErrors.ecOrgNameNotUnique, TypeName(Me), "UpdOrg"
    End If

    'check to see if the pl_ind is changing.
    sSelect = "SELECT pl_ind FROM org WHERE org_id = " & CStr(alOrgID) & " AND
active_ind = '1'"
    Set oRS = GetSQLRecordset(sSelect)
    If Not oRS.EOF Then
        If oRS("pl_ind") <> asPipelineInd Then
            'it's changing, check to see, based on the new value whether a
manufacturer has
            'subscribed to the pipeline or if members have been associated with it.
            If asPipelineInd = "1" Then
                'changing to pipeline org. this is not allowed if members exist for
this org.
                sSelect = "SELECT usr_id FROM member WHERE org_id = " &

```

```

                                BOrg.cls
CStr(aOrgID)
    Set oRs = GetSQLRecordset(sSelect)
    If Not oRs.EOF Then
        RaiseResError DawnOrgErrors.ecMembersAssociated, TypeName(Me),
"UpdOrg"
    Else
        'delete all org_ship_typ, org_bill_typ and org_ship_cost records
        'exist -- these do not apply to pipeline organizations.
        Set oOrg = CtxCreateObject("DawnOrg.DBOrg")
        Call oOrg.DeleteNonPipelineItems(aOrgID)
        'generate a unique subscription code for the organization, if
necessary
        If Trim(asPipelineSubscriptionCd) = "" Then
            'generate a new subscription code.
            asPipelineSubscriptionCd = Crypt("E", "THEROCK",
CStr(aOrgID)) & CStr(aOrgID)
        End If

        Set oOrg = Nothing
    End If
    ElseIf asPipelineInd = "0" Then
        'changing to user org. this is not allowed if someone has subscribed
to it.
        sSelect = "SELECT org_id, pl_mfr_id FROM org WHERE org_id = " &
CStr(aOrgID)
        Set oRs = GetSQLRecordset(sSelect)
        If Not oRs.EOF Then
            If IsNull(oRs("pl_mfr_id")) = False And oRs("pl_mfr_id") <> 0
Then
                RaiseResError DawnOrgErrors.ecAlreadySubscribed,
TypeName(Me), "UpdOrg"
            End If
        End If
    End If
Else
    RaiseResError DawnOrgErrors.ecOrgHasBeenDeleted, TypeName(Me), "UpdOrg"
End If

'start DB component and call updorg function
Set oOrg = CtxCreateObject("DawnOrg.DBOrg")
lRow = oOrg.UpdOrg( _
    aOrgID, asOrgName, asPipelineInd, alPipelineID, asPipelineSubscriptionCd, _
    asDescription, asAddr1, asAddr2, asCity, asState, asPostalCode, asCountry, _
    asPhone, asFax, asEmailAddr, asWebDiscountIndicator,
asDefaultShipCostIndicator, _
    adMaintDate, asAltImagesIndicator)

'check if the update succeeded - ie rows affected
If lRow <> 1 Then
    RaiseResError DawnOrgErrors.ecUpdateOrgFailed, TypeName(Me), "UpdOrg"
End If

'call the sp_update_price procedure to update prices if the web disc ind has
changed.
If bCallSP = True Then
    sSQL = "SELECT DISTINCT prod_line_id FROM org_avail_part WHERE org_id = " &
-
        CStr(aOrgID)
    Set oRs = GetSQLRecordset(sSQL)
    Do Until oRs.EOF

```

```

                                BOrg.cls
lProdLineID = oRs!prod_line_id
'call the procedure for updating ref table prices
Call RunSP("sp_update_price", MakeIntSPParm("aLong", -1), _
                                MakeIntSPParm("aLong", aOrgID), _
                                MakeIntSPParm("aLong", lProdLineID), _
                                MakeIntSPParm("aLong", -1))
                                ORS.MoveNext
                                Loop
End If

'get an RS for the same data
Set UpdOrg = oOrg.GetOrgByID(aOrgID)

'end trans
CtxSetComplete

'cleanup
Set oOrg = Nothing
Set oRs = Nothing
Exit Function

'error handling
UpdOrgErr:
CtxSetAbort
Set oOrg = Nothing
Set oRs = Nothing
RaiseError TypeName(Me), "UpdOrg"

End Function
=====
' this function deletes an Org for a mfr
' takes mfr ID , Org ID , last maintenance date as arguments
' no returns
=====
'business sub to delete orgs, controls transaction
Public Sub DelOrg(ByVal aOrgID As Long, ByVal adMaintDate As Date)

    On Error GoTo DelOrgErr

    Dim oOrg As DawnOrg.DBOrg
    Dim oMemb As DawnUser.DBMember
    Dim oRs As ADODB.Recordset
    Dim lRow As Long
    Dim sSQL As String

    'set all members in this org to null
    Set oMemb = CtxCreateObject("DawnUser.DBMember")
    sSQL = "SELECT usr_id, mfr_id, org_id FROM member WHERE org_id = " &
CStr(aOrgID)
    Set oRs = GetSQLRecordset(sSQL)
    If Not oRs.EOF Then
        oRs.MoveFirst
        Do Until oRs.EOF
            Call oMemb.Update(oRs("usr_id"), 0)
            oRs.MoveNext
        Loop
    End If

    ' create DB comp and call delOrg
    Set oOrg = CtxCreateObject("DawnOrg.DBOrg")
    lRow = oOrg.DelOrg(aOrgID, adMaintDate)

    'check if row was deleted

```

BOrg.cls

```

If lRow <= 0 Then
    'error - no orgs found for given criteria , so no deletion
    RaiseResError DawnOrgErrors.ecNoOrgFound, TypeName(Me), "DelOrg"
End If
If lRow > 1 Then
    'error - too many orgs were found - so, they were not deleted
    RaiseResError DawnOrgErrors.ecMultipleOrgsFound, TypeName(Me), "DelOrg"
End If

'end trans
CtxSetComplete

'cleanup
Set oOrg = Nothing
Set oMemb = Nothing
Set oRS = Nothing
Exit Sub

'error handling
DelOrgErr:
CtxSetAbort
Set oOrg = Nothing
Set oMemb = Nothing
Set oRS = Nothing

RaiseError TypeName(Me), "DelOrg"

End Sub
'=====
' set org ID for all members to null and then deactivate the org
' takes mfr ID , Org ID , last maintenance date as arguments
' no returns
'=====
'business sub to delete orgs, controls transaction
Public Sub DeactivateOrg(ByVal aOrgID As Long, ByVal adMaintDate As Date)

    On Error GoTo ErrHandler

    Dim oOrg As DawnOrg.DBOrg
    Dim oMemb As DawnUser.DBMember
    Dim oRS As ADODB.Recordset
    Dim lRow As Long
    Dim sSQL As String

    'set all members in this org to null
    Set oMemb = CtxCreateObject("DawnUser.DBMember")
    sSQL = "SELECT usr_id, mfr_id, org_id FROM member WHERE org_id = " &
CStr(aOrgID)
    Set oRS = GetSQLRecordset(sSQL)
    If Not oRS.EOF Then
        oRS.MoveFirst
        Do Until oRS.EOF
            Call oMemb.Update(oRS("usr_id"), 0)
            oRS.MoveNext
        Loop
    End If

    ' create DB comp and call delOrg
    Set oOrg = CtxCreateObject("DawnOrg.DBOrg")
    lRow = oOrg.DeactivateOrg(aOrgID, adMaintDate)

    'check if row was deleted
    If lRow <= 0 Then

```

```

                                BOrg.cls
        'error - no orgs found for given criteria , so no deletion
        RaiseResError DawnOrgErrors.ecNoOrgFound, TypeName(Me), "DeactivateOrg"
    End If
    If lRow > 1 Then
        'error - too many orgs were found - so, they were not deleted
        RaiseResError DawnOrgErrors.ecMultipleOrgsFound, TypeName(Me),
"DeactivateOrg"
    End If

    'end trans
    CtxSetComplete

    'cleanup
    Set oOrg = Nothing
    Set oMemb = Nothing
    Set oRs = Nothing
    Exit Sub

    'error handling
ErrorHandler:
    CtxSetAbort
    Set oOrg = Nothing
    Set oMemb = Nothing
    Set oRs = Nothing

    RaiseError TypeName(Me), "DeactivateOrg"
End Sub

'=====
' Reactivates an org
' takes Org ID, last maintenance date as arguments
' no returns
'=====
'business sub to reactivate orgs, controls transaction
Public Sub ReactivateOrg(ByVal aOrgID As Long, ByVal adMaintDate As Date)

    On Error GoTo ErrorHandler

    Dim oOrg As DawnOrg.DBORG
    Dim lRow As Long
    Dim sSQL As String

    ' create DB comp and call delOrg
    Set oOrg = CtxCreateObject("DawnOrg.DBORG")
    lRow = oOrg.ReactivateOrg(aOrgID, adMaintDate)

    'check if row was deleted
    If lRow <= 0 Then
        'error - no orgs found for given criteria , so no reactivation
        RaiseResError DawnOrgErrors.ecNoOrgFound, TypeName(Me), "ReactivateOrg"
    End If
    If lRow > 1 Then
        'error - too many orgs were found - so, they were not reactivated
        RaiseResError DawnOrgErrors.ecMultipleOrgsFound, TypeName(Me),
"ReactivateOrg"
    End If

    'end trans
    CtxSetComplete

    'cleanup
    Set oOrg = Nothing
    Exit Sub

```


BOrg.cls

```

'error handling
ErrorHandler:
  CtxSetAbort
  Set oOrg = Nothing

  RaiseError TypeName(Me), "ReactivateOrg"
End Sub

'=====
' this function creates an Admin for an Org
' takes mfr ID and Org ID as arguments
' returns a record set with Org admin details
'=====
'business function creates a member, sets up access and returns user RS
Public Function CreateOrgAdmin( _
  ByVal alManfID As Long, ByVal alOrgID As Long, ByVal asTitle As String,
  ByVal asFirstName As String, _
  ByVal asMiddleName As String, ByVal asLastName As String, ByVal asEmail As
  String, _
  ByVal asAltEmail As String, ByVal asJobTitle As String, ByVal asPassword As
  String, _
  ByVal asPasswordHint As String, ByVal asChallengePhrase As String, ByVal
  asChallengeResponse As String, _
  ByVal asAddr1 As String, ByVal asAddr2 As String, ByVal asCity As String,
  ByVal asState As String, _
  ByVal asZip As String, ByVal asCountry As String, ByVal asPhone As String,
  ByVal asExtension As String, _
  ByVal asFax As String, ByVal asCompanyName As String, ByVal asAltPhone As
  String, _
  ByVal asAltExr As String) As ADODB.Recordset

  On Error GoTo CreateOrgAdminErr

  'var
  Dim oRs As ADODB.Recordset
  Dim oUser As DawnUser.BMember

  'create a member using the DawnUser component
  Set oUser = CtxCreateObject("DawnUser.BMember")
  Set CreateOrgAdmin = oUser.CreateOrgAdmin( _
    alManfID, alOrgID, asTitle, asFirstName, asMiddleName, asLastName, _
    asEmail, asAltEmail, asJobTitle, asPassword, asPasswordHint, _
    asChallengePhrase, asChallengeResponse, asAddr1, asAddr2, asCity, _
    asState, asZip, asCountry, asPhone, asExtension, asFax,
asCompanyName, _
    asAltPhone, asAltExr)

  'end trans
  CtxSetComplete

  'cleanup
  Set oRs = Nothing
  Set oUser = Nothing
  Exit Function

'error
CreateOrgAdminErr:
  CtxSetAbort
  Set oRs = Nothing
  Set oUser = Nothing
  RaiseError TypeName(Me), "CreateOrgAdmin"

```

BOrg.cls

End Function

```
'=====
'This function updates the USE WEB DISCOUNT INDICATOR
'INPUTS: org ID , web discount indicator, and maintenance date for the Org - not the
'ship address
'RETURNS: none
'7/6/00 chrubecky - added call to sp_update_price function
'=====
```

```
Public Sub SetWebDiscountIndicator(ByVal aOrgID As Long, ByVal
aswebDiscountIndicator As String, _
ByVal adMaintDate As Date)
```

On Error GoTo ErrHandler

```
'var
Dim oDBOrg As DawnOrg.DBOrg
Dim lRow As Long
Dim lProdLineID As Long
Dim sSQL As String
Dim oRS As ADODB.Recordset

'call db comp
Set oDBOrg = CtxCreateObject("DawnOrg.DBOrg")
lRow = oDBOrg.SetWebDiscountIndicator(aOrgID, aswebDiscountIndicator,
adMaintDate)

'check for errors
If lRow <> 1 Then
RaiseResError DawnOrgErrors.ecwebDiscountUpdateFailed, TypeName(Me),
"SetWebDiscountIndicator"
End If
```

```
'call the sp_update_price procedure to update prices.
sSQL = "SELECT DISTINCT prod_line_id FROM org_avail_part WHERE org_id = " & _
CStr(aOrgID)
Set oRS = GetSQLRecordset(sSQL)
Do Until oRS.EOF
lProdLineID = oRS!prod_line_id
'call the procedure for updating ref table prices
Call RunSP("sp_update_price", MakeIntSPParm("aLong", -1), _
MakeIntSPParm("aLong", aOrgID), _
MakeIntSPParm("aLong", lProdLineID), _
MakeIntSPParm("aLong", -1))
oRS.MoveNext
Loop

'cleanup
CtxSetComplete
Set oDBOrg = Nothing
Set oRS = Nothing
Exit Sub
```

```
'error
ErrHandler:
CtxSetAbort
Set oDBOrg = Nothing
Set oRS = Nothing
RaiseError TypeName(Me), "SetWebDiscountIndicator"
```

End Sub

```
'=====
```

```

                                BOrg.cls
'This function updates the USE DEFAULT SHIP COST INDICATOR
'INPUTS: org ID , ship cost indicator, and maintenance date for the Org - not the
ship address
'RETURNS: none
'=====
Public Sub SetDefaultShipCost(ByVal aOrgID As Long, ByVal
asDefaultShipCostIndicator As String, _
                                ByVal adMaintDate As Date)
On Error GoTo ErrHandler

    'var
    Dim oDBOrg As DawnOrg.DBOrg
    Dim lRow As Long

    'call db comp
    Set oDBOrg = CtxCreateObject("DawnOrg.DBOrg")
    lRow = oDBOrg.SetDefaultShipCost(aOrgID, asDefaultShipCostIndicator,
adMaintDate)

    'check for errors
    If lRow <> 1 Then
        RaiseResError DawnOrgErrors.ecUpdateDefaultShipCost, TypeName(Me),
"SetDefaultShipCost"
    End If

    'cleanup
    CtxSetComplete
    Set oDBOrg = Nothing
    Exit Sub

    'error
ErrHandler:
    CtxSetAbort
    Set oDBOrg = Nothing
    RaiseError TypeName(Me), "SetDefaultShipCost"
End Sub

Public Function AddShipType(ByVal aOrgID As Long, ByVal alShipTypeID As Long) As
ADODB.Recordset

    On Error GoTo ErrHandler

    'var
    Dim oDBOrg As DawnOrg.DBOrg
    Dim sSQL As String
    Dim lRows As Long

    'Checks to see if the ship type is already there.
    sSQL = "SELECT * FROM org_avail_ship_typ WHERE org_id = " & aOrgID & " AND
ship_typ_id = " & alShipTypeID
    lRows = RunSQL(sSQL)
    If lRows > 0 Then
        RaiseResError DawnOrgErrors.ecShipIDNotValid, TypeName(Me), "AddShipType"
    End If

    'build business component and call function
    Set oDBOrg = CtxCreateObject("DawnOrg.DBOrg")
    lRows = oDBOrg.AddShipType(aOrgID, alShipTypeID)

    'checks the insert
    If lRows <> 1 Then

```

```

                                BOrg.cls
        RaiseResError DawnOrgErrors.ecInsertShipTypeFailed, TypeName(Me),
"AddShipType"
    End If

    'Returns the added record
    Set AddShipType = GetSQLRecordset(ssSQL)

    'end trans
    CtxSetComplete
    Set oDBOrg = Nothing
    Exit Function

    'error
ErrorHandler:
    CtxSetAbort
    Set oDBOrg = Nothing
    RaiseError TypeName(Me), "AddShipType"
End Function

Public Function AddBillType(ByVal aOrgID As Long, ByVal aBillTypeID As Long) As
ADODB.Recordset

    On Error GoTo ErrorHandler

    'var
    Dim oDBOrg As DawnOrg.DBOrg
    Dim ssSQL As String
    Dim lRows As Long

    'Checks to see if the ship type is already there.
    ssSQL = "SELECT * FROM org_avail_bill_typ WHERE org_id = " & aOrgID & " AND
bill_typ_id = " & aBillTypeID
    lRows = RunSQL(ssSQL)
    If lRows > 0 Then
        RaiseResError DawnOrgErrors.ecBillIDNotValid, TypeName(Me), "AddBillType"
    End If

    'build business component and call function
    Set oDBOrg = CtxCreateObject("DawnOrg.DBOrg")
    lRows = oDBOrg.AddBillType(aOrgID, aBillTypeID)

    'Checks the insert
    If lRows <> 1 Then
        RaiseResError DawnOrgErrors.ecInsertBillTypeFailed, TypeName(Me),
"AddBillType"
    End If

    'Returns the added record
    Set AddBillType = GetSQLRecordset(ssSQL)

    'end trans
    CtxSetComplete
    Set oDBOrg = Nothing
    Exit Function

    'error
ErrorHandler:
    CtxSetAbort
    Set oDBOrg = Nothing
    RaiseError TypeName(Me), "AddBillType"

```

End Function

Public Sub RemoveShipType(ByVal aOrgID As Long, ByVal alShipTypeID As Long)

On Error GoTo ErrHandler

'var

Dim oDBOrg As DawnOrg.DBOrg.

Dim sSQL As String

Dim lRows As Long

'checks to see if the ship type is already there.

sSQL = "SELECT * FROM org_avail_ship_typ WHERE org_id = " & aOrgID & " AND
ship_typ_id = " & alShipTypeID

lRows = RunSQL(sSQL)

If lRows = 0 Then

RaiseResError DawnOrgErrors.ecShipIDNotValid, TypeName(Me), "RemoveShipType"

End If

'build business component and call function

Set oDBOrg = CtxCreateObject("DawnOrg.DBOrg")

oDBOrg.RemoveShipType aOrgID, alShipTypeID

'end trans

CtxSetComplete

Set oDBOrg = Nothing

Exit Sub

'error

ErrHandler:

CtxSetAbort

Set oDBOrg = Nothing

RaiseError TypeName(Me), "RemoveShipType"

End Sub

Public Sub RemoveBillType(ByVal aOrgID As Long, ByVal alBillTypeID As Long)

On Error GoTo ErrHandler

'var

Dim oDBOrg As DawnOrg.DBOrg

Dim sSQL As String

Dim lRows As Long

'checks to see if the ship type is already there.

sSQL = "SELECT * FROM org_avail_bill_typ WHERE org_id = " & aOrgID & " AND
bill_typ_id = " & alBillTypeID

lRows = RunSQL(sSQL)

If lRows = 0 Then

RaiseResError DawnOrgErrors.ecBillIDNotValid, TypeName(Me), "RemoveBillType"

End If

'build business component and call function

Set oDBOrg = CtxCreateObject("DawnOrg.DBOrg")

oDBOrg.RemoveBillType aOrgID, alBillTypeID

'end trans

CtxSetComplete

Set oDBOrg = Nothing

Exit Sub

'error

```

ErrHandler:
    CtxSetAbort
    Set oDBOrg = Nothing
    RaiseError TypeName(Me), "RemoveBillType"
End Sub

Public Sub AddShipTypes(ByVal aOrgID As Long, ByVal arrShipIDs As Variant)
    On Error GoTo ErrHandler

    'var
    Dim oDBShip As DawnOrg.DBOrg
    Dim lRow As Long
    Dim lCounter As Long

    ' create db comp and send result back
    Set oDBShip = CtxCreateObject("DawnOrg.DBOrg")

    'delete ship type
    lRow = oDBShip.DeleteshipType(aOrgID)

    'check for delete success
    If lRow < 0 Then
        RaiseResError DawnOrgErrors.ecDeleteShipTypeFailed, TypeName(Me),
        "DeleteshipType"
    End If

    'add the new ones if there are any in the array
    If Not IsNull(arrShipIDs) Or IsEmpty(arrShipIDs) Then
        If UBound(arrShipIDs) >= 0 Then
            For lCounter = 0 To UBound(arrShipIDs)
                'loop thru all the IDs in the variant array
                lRow = oDBShip.AddShipType(aOrgID, CLng(arrShipIDs(lCounter)))

                'check for insert success
                If lRow <> 1 Then
                    RaiseResError DawnOrgErrors.ecAddShipAddressFailed,
                    TypeName(Me), "AddShipTypes"
                End If
            Next
        End If
    End If

    'cleanup
    CtxSetComplete
    Set oDBShip = Nothing
    Exit Sub

    'error
ErrHandler:
    CtxSetAbort
    Set oDBShip = Nothing
    RaiseError TypeName(Me), "AddShipTypes"
End Sub

Public Sub AddBillTypes(ByVal aOrgID As Long, ByVal arrBillIDs As Variant)
    On Error GoTo ErrHandler

    'var
    Dim oDBOrg As DawnOrg.DBOrg

```

```

                                BOrg.cls

Dim lRow As Long
Dim lCounter As Long

' create db comp and send result back
Set oDBOrg = CtxCreateObject("DawnOrg.DBOrg")

'delete all bill types
lRow = oDBOrg.DeleteBillType(alOrgID)

'check for delete success
If lRow < 0 Then
'        RaiseResError DawnOrgErrors.ecDeleteBillTypeFailed, TypeName(Me),
"DeleteBillType"
End If

'add the new ones if there are any in the array
If Not IsNull(arrBillIDs) Or IsEmpty(arrBillIDs) Then
    If UBound(arrBillIDs) >= 0 Then
        For lCounter = 0 To UBound(arrBillIDs)

            'add row
            lRow = oDBOrg.AddBillType(alOrgID, CLng(arrBillIDs(lCounter)))

            'check for insert success
            If lRow <> 1 Then
                RaiseResError DawnOrgErrors.ecInsertBillTypeFailed,
TypeName(Me), "AddBillTypes"
            End If

        Next
    End If
End If

'cleanup
CtxSetComplete
Set oDBOrg = Nothing
Exit Sub

'error
ErrorHandler:
CtxSetAbort
Set oDBOrg = Nothing
RaiseError TypeName(Me), "AddBillTypes"

End Sub

'=====
'This function validates a pipeline subscription code and adds the subscription if
valid.
'INPUTS: mfr ID , pipeline subscription code, security code
'RETURNS: recordset containing the new pipeline org_id and old pipeline mfr_id (if
applicable)
'chrubeky 6/5/00 - added code to verify mfr is not subscribing to itself
'=====
Public Function PipelineSubscribe(ByVal alManfID As Long, ByVal asSubscrCd As
String, _
                                ByVal aiSecurityCode As SecurityCode) As
ADODB.Recordset
    On Error GoTo ErrorHandler

    Dim sSQL As String
    Dim lOrgID As Long
    Dim lOldOrgID As Long
    Dim lRtn As Long

```

BOrg.cls

```

Dim bSS As Boolean
Dim oRS As ADODB.Recordset
Dim oRSReturn As ADODB.Recordset
Dim oDB As DawnOrg.DBOrg

'verify the subscription code entered is valid, that the org is a pipelining
org, 'no one has subscribed to it already and the manufacturer subscribing is not the
one
'that created the org
ssSQL = "SELECT org_id, mfr_id, org_id as old_org_id, pl_ind, pl_mfr_id,
active_ind FROM org WHERE pl_subscript_cd = '" & _
Trim(asSubscrCd) & "'"
Set oRSReturn = GetSQLRecordset(ssSQL)
If Not oRSReturn.EOF Then
    If oRSReturn.Fields("pl_ind") = "0" Then
        RaiseResError DawnOrgErrors.ecNotPipelineOrg, TypeName(Me),
"PipelineSubscribe"
    ElseIf oRSReturn.Fields("pl_mfr_id") > 0 Then
        RaiseResError DawnOrgErrors.ecMfrAlreadySubscribed, TypeName(Me),
"PipelineSubscribe"
    ElseIf oRSReturn.Fields("active_ind") = "0" Then
        RaiseResError DawnOrgErrors.ecPipelineOrgDeleted, TypeName(Me),
"PipelineSubscribe"
    ElseIf oRSReturn!mfr_id = alManfID Then
        RaiseResError DawnOrgErrors.ecSubscribeToSelf, TypeName(Me),
"PipelineSubscribe"
    Else
        lOrgID = oRSReturn!org_id
        'Make sure the manufacturer can subscribe to the pipeline.
        'Get the ss_ind from the mfr table to see if the mfr is trying to
subscribe to more
        'than the allowed number of sources, if it already has one source,
unsubscribe to
        'that source first.
ssSQL = "SELECT ss_ind FROM mfr WHERE mfr_id = " & CStr(alManfID)
Set oRS = GetSQLRecordset(ssSQL)
If Not oRS.EOF Then
    If oRS.Fields("ss_ind") = "1" Then
        bSS = True
    ElseIf oRS.Fields("ss_ind") = "0" Then
        bSS = False
    End If
    ssSQL = "SELECT org_id as old_org_id FROM org WHERE pl_mfr_id = " &
CStr(alManfID)
    Set oRS = GetSQLRecordset(ssSQL)
    If bSS = True Then
        If Not oRS.EOF Then
            If aiSecurityCode = 1 Then
                lOldOrgID = oRS!old_org_id
                PipelineUnSubscribe alManfID, lOldOrgID,
oRSReturn!mfr_id
            Else
                RaiseResError DawnOrgErrors.ecAlreadyHaveSource,
TypeName(Me), "PipelineSubscribe"
            End If
        End If
    End If
    Set oDB = CtxCreateObject("DawnOrg.DBOrg")
    lRtn = oDB.PipelineSubscribe(alManfID, lOrgID)
    If lRtn > 1 Then
        RaiseResError DawnOrgErrors.ecMultipleOrgsFound, TypeName(Me),
"PipelineSubscribe"
    End If
End If

```



```

                                BOrg.cls
        End If
        ORSReturn!org_id = lOrgID
        ORSReturn!old_org_id = lOldOrgID
        Set PipelineSubscribe = ORSReturn
    Else
        RaiseResError DawnOrgErrors.ecNoMfrType, TypeName(Me),
"PipelineSubscribe"
    End If
End If
Else
    RaiseResError DawnOrgErrors.ecInvalidSubscriptionCd, TypeName(Me),
"PipelineSubscribe"
End If

'cleanup
CtxSetComplete
Set oRs = Nothing
Set oDB = Nothing
Exit Function

ErrHandler:
    CtxSetAbort
    Set oRs = Nothing
    Set oDB = Nothing
    Set PipelineSubscribe = Nothing
    RaiseError TypeName(Me), "PipelineSubscribe"

End Function

'=====
'This function removes a pipeline subscription. also removes parts from the mfr
accept part and all ref tables
'INPUTS: mfr ID
'RETURNS: none
'=====
Private Sub PipelineUnSubscribe(ByVal alManfID As Long, ByVal alOrgID As Long, ByVal
alSrc As Long)
    On Error GoTo ErrHandler

    Dim sSQL As String
    Dim oDB As DawnOrg.DBOrg
    Dim lRtn As Long

    'delete the ref table data for parts
    RunSQL ("DELETE FROM mfr_accept_prod_ref WHERE mfr_id = " & CStr(alManfID) & "
AND src_mfr_id = " & CStr(alSrc))
    RunSQL ("DELETE FROM mfr_accept_part WHERE mfr_id = " & CStr(alManfID) & " AND
src_mfr_id = " & CStr(alSrc))
    RunSQL ("DELETE FROM memb_prod_ref WHERE mfr_id = " & CStr(alManfID))
    RunSQL ("DELETE FROM org_prod_ref WHERE org_id IN ( SELECT org_id FROM org WHERE
mfr_id = " & CStr(alManfID) & ")")

    'delete the subscription. if no subscription exists, no problem.
    Set oDB = CtxCreateObject("DawnOrg.DBOrg")
    lRtn = oDB.PipelineUnSubscribe(alManfID, alOrgID)
    If lRtn <> 1 Then
        RaiseResError DawnOrgErrors.ecPipelineUnSubscribeFailed, TypeName(Me),
"PipelineUnSubscribe"
    End If

    'cleanup
    CtxSetComplete

```

```
Set oDB = Nothing
Exit Sub
```

```
ErrorHandler:
  CtxSetAbort
  Set oDB = Nothing
  RaiseError TypeName(Me), "PipelineUnsubscribe"
End Sub
```

```
'*****
' Description: Encrypts and decrypts a string based on a key.
' Input :   Action - "E" for encrypt, "D" for decrypt.
'           Key    - The key to be used for encrypt or decrypt.
'           Scr    - The string to be encrypted or decrypted.
' Return:   The encrypted or decrypted string.
' Last Checked: 8/12/94
'*****
```

```
Private Function Crypt(ByVal Action As String, ByVal Key As String, ByVal Src As String) As String
```

```
  Dim Count As Integer
  Dim KeyPos As Integer
  Dim KeyLen As Integer
  Dim SrcAsc As Integer
  Dim dest As String
  Dim offset As Integer
  Dim TmpSrcAsc
  Dim SrcPos
```

```
  Key = RTrim$(Key)
  Src = RTrim$(Src)
  Action = UCase(Action)
```

```
  KeyLen = Len(Key)
```

```
  If Action = "E" Then
    Randomize
    offset = (Rnd * 10000 Mod 255) + 1
```

```
    dest = Hex$(offset)
    If Len(dest) = 1 Then
      dest = "0" + dest
    End If
```

```
    For SrcPos = 1 To Len(Src)
      SrcAsc = (Asc(Mid$(Src, SrcPos, 1)) + offset) Mod 255
      If KeyPos < KeyLen Then KeyPos = KeyPos + 1 Else KeyPos = 1
      SrcAsc = SrcAsc Xor Asc(Mid$(Key, KeyPos, 1))
      dest = dest + Format$(Hex$(SrcAsc), "@@")
      offset = SrcAsc
    Next
```

```
    dest = Replace(dest, " ", "I", 1, Len(dest))
  ElseIf Action = "D" Then
    Src = Replace(Src, "I", " ", 1, Len(Src))
    offset = Val("&H" + Left$(Src, 2))
    For SrcPos = 3 To Len(Src) Step 2
      SrcAsc = Val("&H" + Trim(Mid$(Src, SrcPos, 2)))
      If KeyPos < KeyLen Then KeyPos = KeyPos + 1 Else KeyPos = 1
      TmpSrcAsc = SrcAsc Xor Asc(Mid$(Key, KeyPos, 1))
      If TmpSrcAsc <= offset Then
        TmpSrcAsc = 255 + TmpSrcAsc - offset
      Else
        TmpSrcAsc = TmpSrcAsc - offset
      End If
    Next
  End If
```

```
                                BOrg.cls
        End If
        dest = dest + Chr(TmpSrcAsc)
        offset = SrcAsc
    Next
End If

Crypt = dest

End Function
```

BProd.cls

VERSION 1.0 CLASS

BEGIN

MultiUse = -1 'True
Persistable = 0 'NotPersistable
DataBindingBehavior = 0 'vbNone
DataSourceBehavior = 0 'vbNone
MTSTransactionMode = 2 'RequiresTransaction

END

Attribute VB_Name = "BProd"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = True
Option Base 0
Option Explicit

'=====

'Inserts a Product Line into the prod_line table.
'uday 5/30/00 - update mfr tri state code

'=====

Public Function CreateProdLine(ByVal asProdLineNum As String, ByVal asProdLineNme As String, _

ByVal asBriefDescr As String, ByVal asDescr As String, ByVal
asImgUrl As String, _
ByVal alSeq1 As Long, ByVal alSeq2 As Long, ByVal alSeq3 As Long, _
ByVal asLongDescr As String) As ADODB.Recordset
On Error GoTo CreateProdLineErr

Dim lProdLineId As Long
Dim lRtn As Long
Dim oBSeq As DawnSeq.BSequence
Dim oDB As DBProd
Dim sSQL As String
Dim oRS As ADODB.Recordset

'Make sure the prod_line_num is unique.

sSQL = "SELECT COUNT(*) as rtn FROM prod_line WHERE prod_line_num = '" & _
GetSQLStr(asProdLineNum) & "' AND active_ind = '1'"

Set oRS = GetSQLRecordset(sSQL)

If Not oRS.EOF Then

If oRS.Fields("rtn").Value <> 0 Then

RaiseResError DawnProdErrCodes.ecDuplicateProdLineNum, TypeName(Me),
"CreateProdLine(...)"

End If

End If

'Get the next prod_line_id.

Set oBSeq = CtxCreateObject("DawnSeq.BSequence")

lProdLineId = oBSeq.GetNextInSequence("prod_line")

If lProdLineId < 1 Then

RaiseResError DawnProdErrCodes.ecSequenceError, TypeName(Me),
"CreateProdLine(...)"

End If

'Allow the DB component to do the actual insert.

Set oDB = CtxCreateObject("DawnProd.DBProd")

lRtn = oDB.CreateProdLine(lProdLineId, asProdLineNum, asProdLineNme, _
asBriefDescr, asDescr, asImgUrl, alSeq1, alSeq2, alSeq3,
asLongDescr)

If lRtn <> 1 Then 'Something went wrong with the insert

```

        BProd.cls
        RaiseResError DawnProdErrCodes.ecInsertFailed, TypeName(Me),
"CreateProdLine(...)"
    Else
        'Retrieve the newly created record.
        sSQL = "SELECT prod_line_id, prod_line_num, prod_line_nme, brief_descr,
descr," & _
            " img_url, seq_1, seq_2, seq_3, long_descr, last_maint_dtm" & _
            " FROM prod_line WHERE prod_line_id = " & CStr(lProdLineId)

        Set CreateProdLine = GetSQLRecordset(sSQL)
    End If

    'update mfr prod ref
    Call RunSP("sp_mfr_prod_ref_tri_state_update", MakeIntSPParm("aLong",
lProdLineId), _
                                                    MakeIntSPParm("aLong", -1),
-                                                    MakeIntSPParm("aLong", -1))

    'Tell MTS we're done.
    CtxSetComplete

    'Release object references
    Set oBSeq = Nothing
    Set oDB = Nothing
    Set oRS = Nothing

    Exit Function

CreateProdLineErr:
    'Release object references
    Set oBSeq = Nothing
    Set oDB = Nothing
    Set oRS = Nothing

    'set return value

    'Abort the transaction
    CtxSetAbort

    'Raise the error to the calling module
    RaiseError TypeName(Me), "CreateProdLine(...)"

    Exit Function

End Function

'=====
'Performs an update of a specific prod_line record in the database.
'=====
Public Function UpdProdLine(ByVal alProdLineID As Long, ByVal asProdLineNum As
String, _
                        ByVal asProdLineNme As String, ByVal asBriefDescr As String, ByVal
asDescr As String, _
                        ByVal asImgUrl As String, ByVal alSeq1 As Long, ByVal alSeq2 As
Long, _
                        ByVal alSeq3 As Long, ByVal asLongDescr As String, ByVal
adtLastMaintDtm As Date) As ADODB.Recordset
    On Error GoTo UpdProdLineErr

    Dim oDB As DBProd

```

BProd.cls

```

Dim lRtn As Long
Dim sSQL As String
Dim oRS As ADODB.Recordset

'Make sure the prod_line_num is unique.
sSQL = "SELECT Count(*) as rtn FROM prod_line WHERE prod_line_num = '" & _
    & GetSQLStr(asProdLineNum) & "' AND prod_line_id <> '" & _
CStr(alProdLineID) & _
    " AND active_ind = '1' "
Set oRS = GetSQLRecordset(sSQL)
If Not oRS.EOF Then
    If oRS.Fields("rtn").Value <> 0 Then
        RaiseResError DawnProdErrCodes.ecDuplicateProdLineNum, TypeName(Me),
"UpdProdLine(...)"
    End If
End If

'Allow the DB component to do the actual update.
Set oDB = CtxCreateObject("DawnProd.DBProd")
lRtn = oDB.UpdProdLine(alProdLineID, asProdLineNum, asProdLineNme, asBriefDescr,
-
    asDescr, asImgUrl, alSeq1, alSeq2, alSeq3, asLongDescr, adtLastMaintDtm)

' Ensure that a row was actually updated
If lRtn <> 1 Then 'Something went wrong with the update
    sSQL = "SELECT Count(*) as rtn FROM prod_line WHERE prod_line_id = '" & _
        CStr(alProdLineID)
    Set oRS = GetSQLRecordset(sSQL)
    If Not oRS.EOF Then
        If oRS.Fields("rtn").Value = 0 Then
            RaiseResError DawnProdErrCodes.ecRecordDeleted, TypeName(Me),
"UpdProdLineErr(...)"
        Else 'Row changed between retrieve and update.
            RaiseResError DawnProdErrCodes.ecRecordChanged, TypeName(Me),
"UpdProdLineErr(...)"
        End If
    End If
Else
    'Retrieve the newly updated record.
    sSQL = "SELECT prod_line_id, prod_line_num, prod_line_nme, brief_descr,
descr," & _
        " img_url, seq_1, seq_2, seq_3, long_descr, last_maint_dtm" & _
        " FROM prod_line WHERE prod_line_id = '" & CStr(alProdLineID)

    Set UpdProdLine = GetSQLRecordset(sSQL)
End If

'Tell MTS we're done.
CtxSetComplete

'Release object references
Set oDB = Nothing
Set oRS = Nothing

Exit Function

UpdProdLineErr:
'Release object references
Set oDB = Nothing
Set oRS = Nothing

'Set return value

```

BProd.cls

```

'Abort the transaction
CtxSetAbort

'Raise the error to the calling module
RaiseError TypeName(Me), "UpdProdLine(...)"

Exit Function

End Function

'=====
'Deletes a specific prod_line record in the database
'=====
Public Function DelProdLine(ByVal alProdLineID As Long, ByVal adtLastMaintDtm As
Date)
    On Error GoTo DelProdLineErr

    Dim oDB As DBProd
    Dim lRtn As Long
    Dim sSQL As String
    Dim oRS As ADODB.Recordset

    'Allow the DB component to do the actual deletion.
    Set oDB = CtxCreateObject("DawnProd.DBProd")
    lRtn = oDB.DelProdLine(alProdLineID, adtLastMaintDtm)

    If lRtn <> 1 Then 'something went wrong
        sSQL = "SELECT Count(*) as rtn FROM prod_line WHERE prod_line_id = " & _
            CStr(alProdLineID)
        Set oRS = GetSQLRecordset(sSQL)
        If Not oRS.EOF Then
            If oRS.Fields("rtn").Value = 0 Then
                RaiseResError DawnProdErrCodes.ecRecordAlreadyDeleted, TypeName(Me),
"DelProdLine(...)"
            Else 'Row changed between retrieve and delete.
                RaiseResError DawnProdErrCodes.ecRecordChanged, TypeName(Me),
"DelProdLine(...)"
            End If
        End If
    End If

    'Tell MTS the transaction is complete.
    CtxSetComplete

    'Release object references
    Set oDB = Nothing
    Set oRS = Nothing

    Exit Function

DelProdLineErr:
    'Release object references
    Set oDB = Nothing
    Set oRS = Nothing

    'Set return value

    'Abort the transaction
    CtxSetAbort

```

```

                                BProd.cls
'Raise the error to the calling module
RaiseError TypeName(Me), "DelProdLine(...)

Exit Function

End Function

'=====
'deactivates a specific prod_line record in the database
'=====
Public Function DeactivateProdLine(ByVal alProdLineID As Long, ByVal adtLastMaintDtm
AS Date)
    On Error GoTo DeactivateProdLineErr

    Dim oDB As DBProd
    Dim lRtn As Long
    Dim sSQL As String
    Dim oRS As ADODB.Recordset

    'Allow the DB component to do the actual deletion.
    Set oDB = CtxCreateObject("DawnProd.DBProd")
    lRtn = oDB.DeactivateProdLine(alProdLineID, adtLastMaintDtm)

    If lRtn <> 1 Then 'something went wrong
        sSQL = "SELECT Count(*) as rtn FROM prod_line WHERE prod_line_id = " & _
            CStr(alProdLineID)
        Set oRS = GetSQLRecordset(sSQL)
        If Not oRS.EOF Then
            If oRS.Fields("rtn").Value = 0 Then
                RaiseResError DawnProdErrCodes.ecRecordAlreadyDeleted, TypeName(Me),
"DeactivateProdLine(...)"
            Else 'Row changed between retrieve and delete.
                RaiseResError DawnProdErrCodes.ecRecordChanged, TypeName(Me),
"DeactivateProdLine(...)"
            End If
        End If
    End If

    'run stored procedure to update ref tables 1/17/00
    Call RunSP("sp_prod_line_update", MakeIntSPParm("aLong", alProdLineID))

    'Tell MTS the transaction is complete.
    CtxSetComplete

    'Release object references
    Set oDB = Nothing
    Set oRS = Nothing

    Exit Function

DeactivateProdLineErr:
    'Release object references
    Set oDB = Nothing
    Set oRS = Nothing

    'Set return value

    'Abort the transaction
    CtxSetAbort

```



```

                                BProd.cls
'Raise the error to the calling module
RaiseError TypeName(Me), "DeactivateProdLine(...)"

Exit Function

End Function

'=====
'Adds a Commodity to the commodity table.
'uday 5/30/00 update mfr prod ref table for tri states
'=====
Public Function CreateComm(ByVal alProdLineID As Long, ByVal asCommNum As String, _
    ByVal asCommNme As String, ByVal asBriefDescr As String, ByVal asDescr As _
    String, ByVal adbSugListPr As Double, _
    ByVal asImgUrl As String, ByVal alStampface As Long, ByVal alSeq1 As Long,
    ByVal alSeq2 As Long, _
    ByVal alSeq3 As Long, ByVal asLongDescr As String) As ADODB.Recordset
    On Error GoTo CreateCommErr

    Dim lCommId As Long
    Dim lRtn As Long
    Dim oBSeq As DawnSeq.BSequence
    Dim oDB As DBComm
    Dim sSQL As String
    Dim oRS As ADODB.Recordset

    'Make sure the commodity_num is unique.
    sSQL = "SELECT Count(*) as rtn FROM commodity WHERE commodity_num = '" & _
        & GetSQLStr(asCommNum) & "' AND active_ind = '1'"
    Set oRS = GetSQLRecordset(sSQL)
    If Not oRS.EOF Then
        If oRS.Fields("rtn").Value <> 0 Then
            RaiseResError DawnProdErrCodes.ecDuplicateCommNum, TypeName(Me),
"CreateComm(...)"
        End If
    End If

    'Get the next prod_line_id.
    Set oBSeq = CtxCreateObject("DawnSeq.BSequence")
    lCommId = oBSeq.GetNextInSequence("commodity")
    If lCommId < 1 Then
        RaiseResError DawnProdErrCodes.ecSequenceError, TypeName(Me),
"CreateComm(...)"
    End If

    'Allow the DB component to do the actual insert.
    Set oDB = CtxCreateObject("DawnProd.DBComm")
    lRtn = oDB.CreateComm(lCommId, alProdLineID, asCommNum, asCommNme, asBriefDescr,
asDescr, _
        adbSugListPr, asImgUrl, alStampface, alSeq1, alSeq2, alSeq3,
asLongDescr)

    If lRtn <> 1 Then 'Something went wrong with the insert
        RaiseResError DawnProdErrCodes.ecInsertFailed, TypeName(Me),
"CreateComm(...)"
    Else
        'Retrieve the newly created record.
        sSQL = "SELECT commodity_id, prod_line_id, commodity_num, commodity_nme,
brief_descr," & _
            " descr, sugg_list_price, img_url, seq_1, seq_2, seq_3, long_descr," & _

```

```

                                BProd.cls
        " last_maint_dtm FROM commodity WHERE commodity_id = " & CStr(lCommId)

        Set CreateComm = GetSQLRecordset(ssSQL)
    End If

    'update mfr prod ref
    Call RunSP("sp_mfr_prod_ref_tri_state_update", MakeIntSPParm("aLong", -1), _
                                                MakeIntSPParm("aLong", _
lCommId), _
                                                MakeIntSPParm("aLong", -1))

    'Tell MTS we're done.
    CtxSetComplete

    'Release object references
    Set oDB = Nothing
    Set oBSeq = Nothing
    Set oRS = Nothing

    Exit Function

CreateCommErr:
    'Release object references
    Set oDB = Nothing
    Set oBSeq = Nothing
    Set oRS = Nothing

    'Set return value

    'Abort the transaction
    CtxSetAbort

    'Raise the error to the calling module
    RaiseError TypeName(Me), "CreateComm(...)"

    Exit Function

End Function

'=====
'Updates a specific commodity record in the database.
'=====
Public Function UpdComm(ByVal aCommId As Long, ByVal aProdLineID As Long, ByVal
asCommNum As String, _
    ByVal asCommNme As String, ByVal asBriefDescr As String, ByVal asDescr As
String, ByVal adbSugListPr As Double, _
    ByVal asImgUrl As String, ByVal alStampface As Long, ByVal alSeq1 As Long,
ByVal alSeq2 As Long, _
    ByVal alSeq3 As Long, ByVal asLongDescr As String, ByVal adtLastMaintDtm As
Date) As ADODB.Recordset
    On Error GoTo UpdCommErr

    Dim oDB As DBComm
    Dim lRtn As Long
    Dim ssSQL As String
    Dim oRS As ADODB.Recordset

    'Make sure the commodity_num is unique.
    Set oRS = GetSQLRecordset("SELECT Count(*) as rtn FROM commodity " & _
        " WHERE commodity_num = '" & GetSQLStr(asCommNum) & _
        "' AND commodity_id <> " & CStr(aCommId) & " AND active_ind = '1'
Page 7

```

```

")
    If Not oRS.EOF Then
        If oRS.Fields("rtn").Value <> 0 Then
            RaiseResError DawnProdErrCodes.ecDuplicateCommNum, TypeName(Me),
"UpdComm(...)"
            End If
        End If

        'Allow the DB component to do the actual update.
        Set oDB = CtxCreateObject("DawnProd.DBComm")
        lRtn = oDB.UpdComm(alCommId, alProdLineID, asCommNum, asCommNme, asBriefDescr, _
            asDescr, adbSugListPr, asImgUrl, alStampface, alSeq1, alSeq2, alSeq3,
asLongDescr, adtLastMaintDtm)

        If lRtn <> 1 Then 'Something went wrong with the update
            SSQl = "SELECT Count(*) as rtn FROM commodity WHERE commodity_id = " & _
                CStr(alCommId)
            Set oRS = GetSQLRecordset(SSQl)
            If Not oRS.EOF Then
                If oRS.Fields("rtn").Value = 0 Then
                    RaiseResError DawnProdErrCodes.ecRecordDeleted, TypeName(Me),
"UpdComm(...)"
                Else 'Row changed between retrieve and update.
                    RaiseResError DawnProdErrCodes.ecRecordChanged, TypeName(Me),
"UpdComm(...)"
                End If
            End If
        Else
            'Retrieve the newly updated record.

            SSQl = "SELECT commodity_id, prod_line_id, commodity_num, commodity_nme,
brief_descr," & _
                " descr, sugg_list_price, img_url, seq_1, seq_2, seq_3, long_descr," & _
                " last_maint_dtm FROM commodity WHERE commodity_id = " & CStr(alCommId)

            Set UpdComm = GetSQLRecordset(SSQl)
        End If

        'Tell MTS we're done.
        CtxSetComplete

        'Release object references
        Set oDB = Nothing
        Set oRS = Nothing

        Exit Function

UpdCommErr:
    'Release object references
    Set oDB = Nothing
    Set oRS = Nothing

    'Set return value

    'Abort the transaction
    CtxSetAbort

    'Raise the error to the calling module.
    RaiseError TypeName(Me), "UpdComm(...)"

    Exit Function

End Function

```

```

'=====
'Deletes a Commodity from the commodity table.
'=====
Public Function DelComm(ByVal alCommId As Long, ByVal adtLastMaintDtm As Date)
    On Error GoTo DelCommErr

    Dim oDB As DBComm
    Dim lRtn As Long
    Dim sSQL As String
    Dim oRS As ADODB.Recordset

    'Allow the DB component to do the actual deletion.
    Set oDB = CtxCreateObject("DawnProd.DBComm")
    lRtn = oDB.DelComm(alCommId, adtLastMaintDtm)
    If lRtn <> 1 Then 'something went wrong
        sSQL = "SELECT Count(*) as rtn FROM commodity WHERE commodity_id = " & _
            CStr(alCommId)
        Set oRS = GetSQLRecordset(sSQL)
        If Not oRS.EOF Then
            If oRS.Fields("rtn").Value = 0 Then
                RaiseResError DawnProdErrCodes.ecRecordAlreadyDeleted, TypeName(Me),
"DelComm(...)"
            Else 'Row changed between retrieve and delete.
                RaiseResError DawnProdErrCodes.ecRecordChanged, TypeName(Me),
"DelComm(...)"
            End If
        End If
    End If

    'Tell MTS the transaction is complete.
    CtxSetComplete

    'Release object references
    Set oDB = Nothing
    Set oRS = Nothing

    Exit Function

DelCommErr:
    'Release object references
    Set oDB = Nothing
    Set oRS = Nothing

    'Set return value

    'Abort the transaction
    CtxSetAbort

    'Raise the error to the calling module
    RaiseError TypeName(Me), "DelComm(...)"

    Exit Function

End Function

'=====
'deactivates a Commodity from the commodity table.
'=====

```

```

BProd.cls
Public Function DeactivateComm(ByVal alCommId As Long, ByVal adtLastMaintDtm As
Date)
    On Error GoTo DeactivateCommErr

    Dim oDB As DBComm
    Dim lRtn As Long
    Dim sSQL As String
    Dim oRS As ADODB.Recordset

    'Allow the DB component to do the actual deletion.
    Set oDB = CtxCreateObject("DawnProd.DBComm")
    lRtn = oDB.DeactivateComm(alCommId, adtLastMaintDtm)
    If lRtn <> 1 Then 'something went wrong
        sSQL = "SELECT Count(*) as rtn FROM commodity WHERE commodity_id = " & _
            CStr(alCommId)
        Set oRS = GetSQLRecordset(sSQL)
        If Not oRS.EOF Then
            If oRS.Fields("rtn").Value = 0 Then
                RaiseResError DawnProdErrCodes.ecRecordAlreadyDeleted, TypeName(Me),
                "DeactivateComm(...)"
            Else 'Row changed between retrieve and delete.
                RaiseResError DawnProdErrCodes.ecRecordChanged, TypeName(Me),
                "DeactivateComm(...)"
            End If
        End If
    End If

    'run stored procedure to update ref tables 1/17/00
    Call RunSP("sp_commodity_update", MakeIntSPParam("aLong", alCommId))

    'Tell MTS the transaction is complete.
    CtxSetComplete

    'Release object references
    Set oDB = Nothing
    Set oRS = Nothing
    Exit Function

DeactivateCommErr:
    'Release object references
    Set oDB = Nothing
    Set oRS = Nothing

    'Set return value

    'Abort the transaction
    CtxSetAbort

    'Raise the error to the calling module
    RaiseError TypeName(Me), "DeactivateComm(...)"

    Exit Function

End Function

```

```

'=====
'Adds a Part to the part table.
'uday 5/30/00 - update mfr prod ref for new part
'=====
Public Function CreatePart(ByVal alCommId As Long, ByVal asPartNum As String, _

```

```

                                BProd.cls
    ByVal asIntPartNum As String, ByVal asPartNme As String, ByVal asBriefDescr
As String, ByVal asDescr As String, _
    ByVal adbSugListPr As Double, ByVal asImgUrl As String, ByVal asColor As
String, _
    ByVal alSeq1 As Long, ByVal alSeq2 As Long, ByVal alSeq3 As Long, ByVal
asLongDescr As String) _
    As ADODB.Recordset
    On Error GoTo CreatePartErr

    Dim lPartId As Long
    Dim lRtn As Long
    Dim oBSeq As DawnSeq.BSequence
    Dim oDB As DBPart
    Dim sSQL As String
    Dim oRS As ADODB.Recordset

    'Make sure the part_num is unique.
    sSQL = "SELECT Count(*) as rtn FROM part WHERE part_num = '"
        & GetSQLStr(asPartNum) & "' AND active_ind = '1' "
    Set oRS = GetSQLRecordset(sSQL)
    If Not oRS.EOF Then
        If oRS.Fields("rtn").Value <> 0 Then
            RaiseResError DawnProdErrCodes.ecDuplicatePartNum, TypeName(Me),
"CreatePart(...)"
        End If
    End If

    'Get the next prod_line_id.
    Set oBSeq = CtxCreateObject("DawnSeq.BSequence")
    lPartId = oBSeq.GetNextInSequence("part")
    If lPartId < 1 Then
        RaiseResError DawnProdErrCodes.ecSequenceError, TypeName(Me),
"CreatePart(...)"
    End If

    'Allow the DB component to do the actual insert.
    Set oDB = CtxCreateObject("DawnProd.DBPart")
    lRtn = oDB.CreatePart(lPartId, alCommId, asPartNum, asIntPartNum, asPartNme, _
        asBriefDescr, asDescr, adbSugListPr, asImgUrl, asColor, alSeq1,
alseq2, alSeq3, asLongDescr)

    If lRtn <> 1 Then 'Something went wrong with the insert
        RaiseResError DawnProdErrCodes.ecInsertFailed, TypeName(Me),
"CreatePart(...)"
    Else
        'Retrieve the newly created record.
        sSQL = "SELECT part_id, commodity_id, part_num, int_part_num, part_nme,
brief_descr," & _
            " descr, sugg_list_price, img_url, seq_1, seq_2, seq_3, long_descr," & _
            " last_maint_dtm FROM part WHERE part_id = " & CStr(lPartId)

        Set CreatePart = GetSQLRecordset(sSQL)
    End If

    'update mfr prod ref
    Call RunSP("sp_mfr_prod_ref_tri_state_update", MakeIntSPParm("aLong", -1), _
        MakeIntSPParm("aLong", -1),
        _
        MakeIntSPParm("aLong",
lPartId))

    'Tell MTS we're done.
    CtxSetComplete

```

BProd.cls

```

'Release object references
Set oDB = Nothing
Set oBSeq = Nothing
Set oRS = Nothing

Exit Function

CreatePartErr:
'Release object references
Set oDB = Nothing
Set oBSeq = Nothing
Set oRS = Nothing

'Set return value

'Abort the transaction
CtxSetAbort

'Raise the error to the calling module
RaiseError TypeName(Me), "CreatePart(...)"

Exit Function

End Function

'=====
'Updates a specific part record in the database.
'=====
Public Function UpdPart(ByVal alPartID As Long, ByVal alCommId As Long, ByVal
asPartNum As String, _
    ByVal asIntPartNum As String, ByVal asPartNme As String, ByVal asBriefDescr
As String, ByVal asDescr As String, _
    ByVal adbSugListPr As Double, ByVal asImgUrl As String, ByVal asColor As
String, _
    ByVal alSeq1 As Long, ByVal alSeq2 As Long, ByVal alSeq3 As Long, ByVal
asLongDescr As String, _
    ByVal adtLastMaintDtm As Date) As ADODB.Recordset
    On Error GoTo UpdPartErr

    Dim oDB As DBPart
    Dim lRtn As Long
    Dim sSQL As String
    Dim oRS As ADODB.Recordset

    'Make sure the part_num is unique.
    sSQL = "SELECT Count(*) as rtn FROM part WHERE part_num = '" _
        & GetSQLStr(asPartNum) & "' AND part_id <> " & CStr(alPartID) & _
        " AND active_ind = '1' "
    Set oRS = GetSQLRecordset(sSQL)
    If Not oRS.EOF Then
        If oRS.Fields("rtn").Value <> 0 Then
            RaiseResError DawnProdErrCodes.ecDuplicatePartNum, TypeName(Me),
"CreatePart(...)"
        End If
    End If

    'Allow the DB component to do the actual update.
    Set oDB = CtxCreateObject("DawnProd.DBPart")
    lRtn = oDB.UpdPart(alPartID, alCommId, asPartNum, asIntPartNum, asPartNme,
asBriefDescr, _

```

```

                                BProd.cls
    asDescr, adbSugListPr, asImgUrl, asColor, alSeq1, alSeq2, alSeq3,
asLongDescr, adtLastMaintDtm)

    If lRtn <> 1 Then 'Something went wrong with the update
        sSQL = "SELECT Count(*) as rtn FROM part WHERE part_id = " & _
                CStr(alPartID)
        Set oRS = GetSQLRecordset(sSQL)
        If Not oRS.EOF Then
            If oRS.Fields("rtn").Value = 0 Then
                RaiseResError DawnProdErrCodes.ecRecordDeleted, TypeName(Me),
"UpdPart(...)"
            Else 'Row changed between retrieve and update.
                RaiseResError DawnProdErrCodes.ecRecordChanged, TypeName(Me),
"UpdPart(...)"
            End If
        End If
    Else
        'Retrieve the newly updated record.
        sSQL = "SELECT part_id, commodity_id, part_num, int_part_num, part_nme,
brief_descr," & _
            " descr, sugg_list_price, img_url, seq_1, seq_2, seq_3, long_descr," & _
            " last_maint_dtm FROM part WHERE part_id = " & CStr(alPartID)

        Set UpdPart = GetSQLRecordset(sSQL)
    End If

    'Tell MTS we're done.
    CtxSetComplete

    'Release object references
    Set oDB = Nothing
    Set oRS = Nothing

    Exit Function

UpdPartErr:
    'Release object references
    Set oDB = Nothing
    Set oRS = Nothing

    'Set return value

    'Abort the transaction
    CtxSetAbort

    'Raise the error to the calling module
    RaiseError TypeName(Me), "UpdPart(...)"

    Exit Function

End Function

'=====
'Deletes a specific part record in the database.
'=====

Public Function DelPart(ByVal alPartID As Long, ByVal adtLastMaintDtm As Date)
    On Error GoTo DelPartErr

    Dim oDB As DBPart
    Dim lRtn As Long

```


BProd.cls

```

Dim sSQL As String
Dim oRS As ADODB.Recordset

'Allow the DB component to do the actual deletion.
Set oDB = CtxCreateObject("DawnProd.DBPart")
lRtn = oDB.DelPart(alPartID, adtLastMaintDtm)

If lRtn <> 1 Then 'something went wrong
    sSQL = "SELECT Count(*) as rtn FROM part WHERE part_id = " & _
        CStr(alPartID)
    Set oRS = GetSQLRecordset(sSQL)
    If Not oRS.EOF Then
        If oRS.Fields("rtn").Value = 0 Then
            RaiseResError DawnProdErrCodes.ecRecordAlreadyDeleted, TypeName(Me),
"DelPart(...)"
        Else 'Row changed between retrieve and delete.
            RaiseResError DawnProdErrCodes.ecRecordChanged, TypeName(Me),
"DelPart(...)"
        End If
    End If
End If

'Tell MTS the transaction is complete.
CtxSetComplete

'Release object references
Set oDB = Nothing
Set oRS = Nothing

Exit Function

DelPartErr:
'Release object references
Set oDB = Nothing
Set oRS = Nothing

'Set return value

'Abort the transaction
CtxSetAbort

'Raise the error to the calling module
RaiseError TypeName(Me), "DelPart(...)"

Exit Function

End Function

'=====
'deactivates a specific part record in the database.
'=====

Public Function DeactivatePart(ByVal alPartID As Long, ByVal adtLastMaintDtm As
Date)
    On Error GoTo DeactivatePartErr

    Dim oDB As DBPart
    Dim lRtn As Long
    Dim sSQL As String
    Dim oRS As ADODB.Recordset

    'Allow the DB component to do the actual deletion.

```

```

        BProd.cls
Set oDB = CtxCreateObject("DawnProd.DBPart")
lRtn = oDB.DeactivatePart(alPartID, adtLastMaintDtm)

If lRtn <> 1 Then 'something went wrong
    sSQL = "SELECT Count(*) as rtn FROM part WHERE part_id = " & _
        CStr(alPartID)
    Set oRS = GetSQLRecordset(sSQL)
    If Not oRS.EOF Then
        If oRS.Fields("rtn").Value = 0 Then
            RaiseResError DawnProdErrCodes.ecRecordAlreadyDeleted, TypeName(Me),
"DeactivatePart(...)"
        Else 'Row changed between retrieve and delete.
            RaiseResError DawnProdErrCodes.ecRecordChanged, TypeName(Me),
"DeactivatePart(...)"
        End If
    End If
End If

'run stored procedure to update ref tables 1/17/00
Call RunSP("sp_part_update", MakeIntSPParm("aLong", alPartID))

'Tell MTS the transaction is complete.
CtxSetComplete

'Release object references
Set oDB = Nothing
Set oRS = Nothing

Exit Function

DeactivatePartErr:
'Release object references
Set oDB = Nothing
Set oRS = Nothing

'Set return value

'Abort the transaction
CtxSetAbort

'Raise the error to the calling module
RaiseError TypeName(Me), "DeactivatePart(...)"

Exit Function

End Function

'=====
'Adds a Category record in the category table.
'=====

Public Function CreateCat(ByVal asCatNme As String, ByVal asImgUrl As String, _
    ByVal asBriefDescr As String, ByVal asDescr As String) As ADODB.Recordset
    On Error GoTo CreateCatErr

    Dim lCatId As Long
    Dim lRtn As Long
    Dim oBSeq As DawnSeq.BSequence
    Dim oDB As DBCat
    Dim sSQL As String
    Dim oRS As ADODB.Recordset

```

BProd.cls

```

'Make sure the cat_nme is unique.
ssSQL = "SELECT Count(*) as rtn FROM category WHERE cat_nme = '" & _
        & GetSQLStr(asCatNme) & "' AND active_ind = '1' " & _
Set oRS = GetSQLRecordset(ssSQL)
If Not oRS.EOF Then
    If oRS.Fields("rtn").Value <> 0 Then
        RaiseResError DawnProdErrCodes.ecDuplicateCatNme, TypeName(Me),
"CreateCat(...)"
    End If
End If

'Get the next cat_id.
Set oBSeq = CtxCreateObject("DawnSeq.BSequence")
lCatId = oBSeq.GetNextInSequence("category")
If lCatId < 1 Then
    RaiseResError DawnProdErrCodes.ecSequenceError, TypeName(Me),
"CreateCat(...)"
End If

'Allow the DB component to do the actual insert.
Set oDB = CtxCreateObject("DawnProd.DBCat")
lRtn = oDB.CreateCat(lCatId, asCatNme, asImgUrl, asBriefDescr, asDescr)

If lRtn <> 1 Then 'Something went wrong with the insert
    RaiseResError DawnProdErrCodes.ecInsertFailed, TypeName(Me),
"CreateCat(...)"
Else
    'Retrieve the newly created record.
    ssSQL = "SELECT cat_id, cat_nme, brief_descr, descr, last_maint_dtm" & _
            " FROM category WHERE cat_id = " & CStr(lCatId)

    Set CreateCat = GetSQLRecordset(ssSQL)
End If

'Tell MTS we're done.
CtxSetComplete

'Release object references
Set oDB = Nothing
Set oBSeq = Nothing
Set oRS = Nothing

Exit Function

CreateCatErr:
'Release object references
Set oDB = Nothing
Set oBSeq = Nothing
Set oRS = Nothing

'Set return value

'Abort the transaction
CtxSetAbort

'Raise the error to the calling module
RaiseError TypeName(Me), "CreateCat(...)"

Exit Function

End Function

```

BProd.cls

'=====

'Updates a specific Category record in the category table.

'=====

Public Function UpdCat(ByVal alCatId As Long, ByVal asCatNme As String, _
ByVal asImgUrl As String, ByVal asBriefDescr As String, ByVal asDescr As
String, _
ByVal adtLastMaintDtm As Date) As ADODB.Recordset
On Error GoTo UpdCatErr

Dim oDB As DBCat
Dim lRtn As Long
Dim sSQL As String
Dim oRS As ADODB.Recordset

'Make sure the cat_nme is unique.
sSQL = "SELECT Count(*) as rtn FROM category WHERE cat_nme = '" _
& GetSQLStr(asCatNme) & "' AND cat_id <> " & CStr(alCatId) & " AND
active_ind = '1' "
Set oRS = GetSQLRecordset(sSQL)
If Not oRS.EOF Then
If oRS.Fields("rtn").Value <> 0 Then
RaiseResError DawnProdErrCodes.ecDuplicateCatNme, TypeName(Me),
"UpdCat(...)"
End If
End If

'Allow the DB component to do the actual update.
Set oDB = CtxCreateObject("DawnProd.DBCat")
lRtn = oDB.UpdCat(alCatId, asCatNme, asImgUrl, asBriefDescr, asDescr,
adtLastMaintDtm)

If lRtn <> 1 Then 'Something went wrong with the update
sSQL = "SELECT Count(*) as rtn FROM category WHERE cat_id = " & _
CStr(alCatId)
Set oRS = GetSQLRecordset(sSQL)
If Not oRS.EOF Then
If oRS.Fields("rtn").Value = 0 Then 'The record doesn't exist.
RaiseResError DawnProdErrCodes.ecRecordDeleted, TypeName(Me),
"UpdCat(...)"
Else 'Row changed between retrieve and update.
RaiseResError DawnProdErrCodes.ecRecordChanged, TypeName(Me),
"UpdCat(...)"
End If
End If

Else
'Retrieve the newly updated record.
sSQL = "SELECT cat_id, cat_nme, brief_descr, descr, last_maint_dtm" & _
" FROM category WHERE cat_id = " & CStr(alCatId)

Set UpdCat = GetSQLRecordset(sSQL)
End If

'Tell MTS we're done.
CtxSetComplete

'Release object references
Set oDB = Nothing
Set oRS = Nothing

Exit Function

UpdCatErr:

'Release object references
Set oDB = Nothing
Set oRS = Nothing

'Set return value

'Abort the transaction
CtxSetAbort

'Raise the error to the calling module
RaiseError TypeName(Me), "UpdCat(...)"

Exit Function

End Function

'=====

'Deletes a specific Category record from the category table.
'it also deletes all association with product lines in the prod_line_cat table
'=====

Public Function DelCat(ByVal alCatId As Long, ByVal adtLastMaintDtm As Date)
On Error GoTo DelCatErr

Dim oDB As DBCat
Dim oDBProdLineCat As DBProdLineCat
Dim lRtn As Long
Dim oRS As ADODB.Recordset
Dim sSQL As String

'delete the association with prodlines in the prod_line Cat table
'multiple rows can be deleted here or no rows because, there might not be any
associations

'so we check for < 0 only
Set oDBProdLineCat = CtxCreateObject("DawnProd.DBProdLineCat")
lRtn = oDBProdLineCat.RemProdLinesInCat(alCatId)
If lRtn < 0 Then
RaiseResError DawnProdErrCodes.ecRecordAlreadyDeleted, TypeName(Me),
"DelCat(...)"
End If

'Allow the DB component to do the actual deletion.
Set oDB = CtxCreateObject("DawnProd.DBCat")
lRtn = oDB.DelCat(alCatId, adtLastMaintDtm)

'something went wrong
If lRtn <> 1 Then
sSQL = "SELECT Count(*) as rtn FROM category WHERE cat_id = " & _
CStr(alCatId)
Set oRS = GetSQLRecordset(sSQL)
If Not oRS.EOF Then
If oRS.Fields("rtn").Value = 0 Then
RaiseResError DawnProdErrCodes.ecRecordAlreadyDeleted, TypeName(Me),
"DelCat(...)"
Else 'Row changed between retrieve and delete.
RaiseResError DawnProdErrCodes.ecRecordChanged, TypeName(Me),
"DelCat(...)"
End If
End If
End If

```

                                BProd.cls
'Tell MTS the transaction is complete.
CtxSetComplete

'Release object references
Set oDB = Nothing
Set oRS = Nothing
Set oDBProdLineCat = Nothing

Exit Function

DelCatErr:
'Release object references
Set oDB = Nothing
Set oRS = Nothing
Set oDBProdLineCat = Nothing

'Set return value

'Abort the transaction
CtxSetAbort

'Raise the error to the calling module
RaiseError TypeName(Me), "DelCat(...)"

Exit Function

End Function

'=====
'deactivates a specific Category record from the category table.
'it also deletes all association with product lines in the prod_line_cat table
'=====
Public Function DeactivateCat(ByVal alCatId As Long, ByVal adtLastMaintDtm As Date)
    On Error GoTo DeactivateCatErr

    Dim oDB As DBCat
    Dim oDBProdLineCat As DBProdLineCat
    Dim lRtn As Long
    Dim oRS As ADODB.Recordset
    Dim sSQL As String

    'create DB component
    Set oDB = CtxCreateObject("DawnProd.DBCat")
    Set oDBProdLineCat = CtxCreateObject("DawnProd.DBProdLineCat")

    'delete the association with prodlines in the prod_line Cat table
    'multiple rows can be deleted here or no rows because, there might not be any
associations
    'so we check for < 0 only
    lRtn = oDBProdLineCat.RemProdLinesInCat(alCatId)
    If lRtn < 0 Then
        RaiseResError DawnProdErrCodes.ecRecordAlreadyDeleted, TypeName(Me),
"DeactivateCat(...)"
    End If

    'Allow the DB component to do the actual deletion.
    lRtn = oDB.DeactivateCat(alCatId, adtLastMaintDtm)

    'something went wrong
    If lRtn <> 1 Then
        sSQL = "SELECT Count(*) as rtn FROM category WHERE cat_id = " & _

```

BProd.cls

```

        CStr(alCatId)
    Set oRS = GetSQLRecordset(ssQL)
    If Not oRS.EOF Then
        If oRS.Fields("rtn").Value = 0 Then
            RaiseResError DawnProdErrCodes.ecRecordAlreadyDeleted, TypeName(Me),
"DeactivateCat(...)"
        Else 'Row changed between retrieve and delete.
            RaiseResError DawnProdErrCodes.ecRecordChanged, TypeName(Me),
"DeactivateCat(...)"
        End If
    End If
End If

'Tell MTS the transaction is complete.
CtxSetComplete

'Release object references
Set oDB = Nothing
Set oRS = Nothing
Set oDBProdLineCat = Nothing

Exit Function

```

```

DeactivateCatErr:
'Release object references
Set oDB = Nothing
Set oRS = Nothing
Set oDBProdLineCat = Nothing

'Set return value

'Abort the transaction
CtxSetAbort

'Raise the error to the calling module
RaiseError TypeName(Me), "DeactivateCat(...)"

Exit Function

```

End Function

```

'=====
'Adds a relationship between a product line and a category.
'changed 10/21/99 - uday - deletes all relations and adds all the ones in the array
passed in
'returns number of rows affected
'=====

```

```

Public Function AddProdLineCat(ByVal alCatId As Long, ByVal arrProdLine As Variant)
As Long

```

```

    On Error GoTo AddProdLineCatErr

```

```

    Dim oDB As DBProdLineCat
    Dim lRtn As Long
    Dim oRS As ADODB.Recordset
    Dim ssQL As String
    Dim lCounter As Long

```

```

'create DB component
Set oDB = CtxCreateObject("DawnProd.DBProdLineCat")

```

```

                                BProd.cls
'get all the prodline cats for the given cat
sSQL = "SELECT cat_id, prod_line_id FROM prod_line_cat WHERE cat_id = " &
CStr(alCatId)
Set oRS = GetSQLRecordset(sSQL)

'delete all the relations
If Not oRS.EOF Then
oRS.MoveFirst
Do Until oRS.EOF
    lRtn = oDB.RemProdLineCat(alCatId, oRS("prod_line_id"))
    If lRtn <> 1 Then
        RaiseResError DawnProdErrCodes.ecDeleteFailed, TypeName(Me),
"AddProdLineCat(...)"
    End If
    oRS.MoveNext
Loop
End If

'check the array. If there are no elements, then dont add any relations , else
add all of them
If Not (IsEmpty(arrProdLine) Or UBound(arrProdLine) < 0) Then
    For lCounter = 0 To UBound(arrProdLine)

        'Allow the DB component to do the actual insert.
        lRtn = oDB.AddProdLineCat(alCatId, arrProdLine(lCounter))
        If lRtn <> 1 Then
            RaiseResError DawnProdErrCodes.ecInsertFailed, _
                TypeName(Me), "AddProdLineCat(...)"
        End If

    Next

End If

AddProdLineCat = lRtn

'Tell MTS the transaction is complete.
CtxSetComplete

'Release object references
Set oDB = Nothing
Set oRS = Nothing
Exit Function

AddProdLineCatErr:
'Release object references
Set oDB = Nothing
Set oRS = Nothing
'Set return value
AddProdLineCat = -1

'Abort the transaction
CtxSetAbort

'Raise the error to the calling module
RaiseError TypeName(Me), "AddProdLineCat(...)"

Exit Function

End Function

```

```

'=====
=====

```


BProd.cls

'Removes a relationship between a product line and a category.

'=====

Public Function RemProdLineCat(ByVal alCatId As Long, ByVal alProdLineID As Long)
On Error GoTo RemProdLineCatErr

Dim oDB As DBProdLineCat
Dim lRtn As Long
Dim oRS As ADODB.Recordset
Dim sSQL As String

'Allow the DB component to do the actual deletion.
Set oDB = CtxCreateObject("DawnProd.DBProdLineCat")
lRtn = oDB.RemProdLineCat(alCatId, alProdLineID)

If lRtn <> 1 Then 'something went wrong
sSQL = "SELECT Count(*) as rtn FROM prod_line_cat WHERE prod_line_id = " & _
CStr(alProdLineID) & " AND cat_id = " & CStr(alCatId)

Set oRS = GetSQLRecordset(sSQL)

If Not oRS.EOF Then

If oRS.Fields("rtn").value <> 1 Then 'already deleted

RaiseResError DawnProdErrCodes.ecRecordAlreadyDeleted, TypeName(Me),

"RemProdLineCat(...)"

Else

RaiseResError DawnProdErrCodes.ecDeleteFailed, TypeName(Me),

"RemProdLineCat(...)"

End If

End If

End If

RemProdLineCat = lRtn

'Tell MTS the transaction is complete.

CtxSetComplete

'Release object references

Set oDB = Nothing

Set oRS = Nothing

Exit Function

RemProdLineCatErr:

'Release object references

Set oDB = Nothing

Set oRS = Nothing

'Set return value

'Abort the transaction

CtxSetAbort

'Raise the error to the calling module

RaiseError TypeName(Me), "RemProdLineCat(...)"

Exit Function

End Function

'=====

'=====

'Adds a relationship between a commodity and a category.

'=====

'=====

Public Function AddCommCat(ByVal alCatId As Long, ByVal alCommId As Long) As Long
On Error GoTo AddCommCatErr

BProd.cls

```

Dim oDB As DBCommCat
Dim lRtn As Long

'Allow the DB component to do the actual insert.
Set oDB = CtxCreateObject("DawnProd.DBCommCat")
lRtn = oDB.AddCommCat(alCatId, alCommId)
If lRtn <> 1 Then
    RaiseResError DawnProdErrCodes.ecInsertFailed, TypeName(Me),
"AddCommCat(...)"
End If
AddCommCat = lRtn

'Tell MTS the transaction is complete.
CtxSetComplete

'Release object references
Set oDB = Nothing

Exit Function

AddCommCatErr:
'Release object references
Set oDB = Nothing

'Set return value
AddCommCat = -1

'Abort the transaction
CtxSetAbort

'Raise the error to the calling module
RaiseError TypeName(Me), "AddCommCat(...)"

Exit Function

End Function

'=====
'Removes a relationship between a commodity and a category.
'=====
Public Function RemCommCat(ByVal alCatId As Long, ByVal alCommId As Long)
    On Error GoTo RemCommCatErr

    Dim oDB As DBCommCat
    Dim lRtn As Long
    Dim oRS As ADODB.Recordset
    Dim sSQL As String

    'Allow the DB component to do the actual deletion.
    Set oDB = CtxCreateObject("DawnProd.DBCommCat")
    lRtn = oDB.RemCommCat(alCatId, alCommId)
    If lRtn <> 1 Then 'something went wrong
        sSQL = "SELECT Count(*) as rtn FROM comm_cat WHERE category_id = " & _
            CStr(alCommId) & " AND cat_id = " & CStr(alCatId)
        Set oRS = GetSQLRecordset(sSQL)
        If Not oRS.EOF Then
            If oRS.Fields("rtn").Value <> 1 Then 'already deleted
                RaiseResError DawnProdErrCodes.ecRecordAlreadyDeleted, TypeName(Me),
"RemCommCat(...)"
            Else

```

```

                                BProd.cls
                                RaiseResError DawnProdErrCodes.ecDeleteFailed, TypeName(Me),
"RemCommCat(...)"
    End If
    End If
    End If
    RemCommCat = lRtn

```

```

'Tell MTS the transaction is complete.
CtxSetComplete

```

```

'Release object references
Set oDB = Nothing
Set oRS = Nothing

```

```

Exit Function

```

```

RemCommCatErr:
'Release object references
Set oDB = Nothing
Set oRS = Nothing

'Set return value

'Abort the transaction
CtxSetAbort

'Raise the error to the calling module
RaiseError TypeName(Me), "RemCommCat(...)"

Exit Function

```

```

End Function

```

```

'=====
'Adds a relationship between a commodity and a category.
'=====
=====

```

```

Public Function AddPartCat(ByVal aCatId As Long, ByVal aPartID As Long) As Long
    On Error GoTo AddPartCatErr

```

```

    Dim oDB As DBPartCat
    Dim lRtn As Long

```

```

'Allow the DB component to do the actual insert.
Set oDB = CtxCreateObject("DawnProd.DBPartCat")
lRtn = oDB.AddPartCat(aCatId, aPartID)
If lRtn <> 1 Then

```

```

    RaiseResError DawnProdErrCodes.ecInsertFailed, TypeName(Me),
"AddPartCat(...)"
    End If
    AddPartCat = lRtn

```

```

'Tell MTS the transaction is complete.
CtxSetComplete

```

```

'Release object references
Set oDB = Nothing

```

```

Exit Function

```

```

AddPartCatErr:
'Release object references

```

```

Set oDB = Nothing

'Set return value
AddPartCat = -1

'Abort the transaction
CtxSetAbort

'Raise the error to the calling module
RaiseError TypeName(Me), "AddPartCat(...)"

Exit Function

End Function

'=====
'Removes a relationship between a part and a category.
'=====
Public Function RemPartCat(ByVal alCatId As Long, ByVal alPartID As Long)
    On Error GoTo RemPartCatErr

    Dim oDB As DBPartCat
    Dim lRtn As Long
    Dim oRS As ADODB.Recordset
    Dim sSQL As String

    'Allow the DB component to do the actual deletion.
    Set oDB = CtxCreateObject("DawnProd.DBPartCat")
    lRtn = oDB.RemPartCat(alCatId, alPartID)
    If lRtn <> 1 Then 'something went wrong
        sSQL = "SELECT Count(*) as rtn FROM part_cat WHERE part_id = " & _
            CStr(alPartID) & " AND cat_id = " & CStr(alCatId)
        Set oRS = GetSQLRecordset(sSQL)
        If Not oRS.EOF Then
            If oRS.Fields("rtn").Value <> 1 Then 'already deleted
                RaiseResError DawnProdErrCodes.ecRecordAlreadyDeleted, TypeName(Me),
"RemPartCat(...)"
            Else
                RaiseResError DawnProdErrCodes.ecDeleteFailed, TypeName(Me),
"RemPartCat(...)"
            End If
        End If
    End If
    RemPartCat = lRtn

    'Tell MTS the transaction is complete.
    CtxSetComplete

    'Release object references
    Set oDB = Nothing
    Set oRS = Nothing

    Exit Function

RemPartCatErr:
    'Release object references
    Set oDB = Nothing
    Set oRS = Nothing

    'Set return value

```

```

'Abort the transaction
CtxSetAbort

'Raise the error to the calling module
RaiseError TypeName(Me), "RemPartCat(...)"

Exit Function

End Function

'=====
'Adds pricing information for a product line associated with a manufacturer.
'2000/01/28 chrubecky added extra transaction complete before stored procedure
call.
'=====
Public Function CreateManfProdLinePr(ByVal alManfID As Long, ByVal alProdLineID As
Long, _
ByVal adbWebDiscVal As Double, ByVal asWebDiscTypCd As String) As
ADODB.Recordset
On Error GoTo CreateManfProdLinePrErr

Dim oDB As DBProdLinePricing
Dim lRtn As Long
Dim sSQL As String

asWebDiscTypCd = Trim(asWebDiscTypCd)
If asWebDiscTypCd <> "" Then
If asWebDiscTypCd <> CStr(DiscountTypeCode.ecDollar) And _
asWebDiscTypCd <> CStr(DiscountTypeCode.ecPercentage) Then
RaiseResError DawnProdErrCodes.ecInvalidDiscTypCd, TypeName(Me),
"CreateManfProdLinePr(...)"
ElseIf adbWebDiscVal < 0 Then
RaiseResError DawnProdErrCodes.ecWebDiscountValueRequired, TypeName(Me),
-
"CreateManfProdLinePr(...)"
End If
End If

'Allow the DB component to do the actual insert.
Set oDB = CtxCreateObject("DawnProd.DBProdLinePricing")
lRtn = oDB.CreateManfProdLinePr(alManfID, alProdLineID, adbWebDiscVal,
asWebDiscTypCd)
If lRtn <> 1 Then 'Something went wrong on the insert.
RaiseResError DawnProdErrCodes.ecInsertFailed, TypeName(Me),
"CreateManfProdLinePr(...)"
Else
'Now get the newly created record.
sSQL = "SELECT mfr_id, prod_line_id, web_disc_typ_cd, web_disc_val," & _
" last_maint_dtm FROM mfr_prod_line_pricing WHERE mfr_id = " & _
CStr(alManfID) & _
" AND prod_line_id = " & CStr(alProdLineID)
Set CreateManfProdLinePr = GetSQLRecordset(sSQL)
End If

'Tell MTS the transaction is complete.
CtxSetComplete

'Release object references
Set oDB = Nothing

Exit Function

```

BProd.cls

```

CreateManfProdLinePrErr:
'Release object references
Set oDB = Nothing

'Set return value

'Abort the transaction
CtxSetAbort

'Raise the error to the calling module
RaiseError TypeName(Me), "CreateManfProdLinePr(...)"

Exit Function

End Function

'=====
'Updates the pricing information for a product line associated with a manufacturer.
'2000/01/28 chrubecky added extra transaction complete before stored procedure
call.
'=====
Public Function UpdManfProdLinePr(ByVal alManfID As Long, ByVal alProdLineID As
Long, _
    ByVal adbwebDiscVal As Double, ByVal aswebDiscTypCd As String, _
    ByVal adtLastMaintDtm As Date) As ADODB.Recordset
On Error GoTo UpdManfProdLinePrErr

Dim oDB As DBProdLinePricing
Dim lRtn As Long
Dim sSQL As String
Dim oRS As ADODB.Recordset

'If a discount type code was entered, validate it.
aswebDiscTypCd = Trim(aswebDiscTypCd)
If aswebDiscTypCd <> "" Then
    If aswebDiscTypCd <> CStr(DiscountTypeCode.ecDollar) And _
        aswebDiscTypCd <> CStr(DiscountTypeCode.ecPercentage) Then
        RaiseResError DawnProdErrCodes.ecInvalidDiscTypCd, TypeName(Me),
"UpdManfProdLinePr(...)"
    ElseIf adbwebDiscVal < 0 Then
        RaiseResError DawnProdErrCodes.ecWebDiscountValueRequired, TypeName(Me),
-
        "UpdManfProdLinePr(...)"
    End If
End If

'Allow the DB component to do the actual update.
Set oDB = CtxCreateObject("DawnProd.DBProdLinePricing")
lRtn = oDB.UpdManfProdLinePr(alManfID, alProdLineID, adbwebDiscVal,
aswebDiscTypCd, _
    adtLastMaintDtm)
If lRtn <> 1 Then 'Something went wrong on the update.
    sSQL = "SELECT Count(*) as rtn FROM mfr_prod_line_pricing WHERE mfr_id = " &
CStr(alManfID) & _
        " AND prod_line_id = " & CStr(alProdLineID)
    Set oRS = GetSQLRecordset(sSQL)
    If Not oRS.EOF Then
        If oRS.Fields("rtn").Value = 0 Then
            RaiseResError DawnProdErrCodes.ecRecordAlreadyDeleted, TypeName(Me),
"DelCat(...)"

```

```

                                BProd.cls
        Else 'Row changed between retrieve and delete.
            RaiseResError DawnProdErrCodes.ecRecordChanged, TypeName(Me),
"DelCat(...)"
        End If
    End If
Else
    'Now get the newly updated record.
    sSQL = "SELECT mfr_id, prod_line_id, web_disc_typ_cd, web_disc_val," & _
        " last_maint_dtm FROM mfr_prod_line_pricing WHERE mfr_id = " & _
CStr(alManfID) & _
        " AND prod_line_id = " & CStr(alProdLineID)
    Set UpdManfProdLinePr = GetSQLRecordset(sSQL)
End If

'Tell MTS the transaction is complete.
CtxSetComplete

'Release object references
Set ODB = Nothing
Set ORS = Nothing

```

Exit Function

```

UpdManfProdLinePrErr:
'Release object references
Set ODB = Nothing
Set ORS = Nothing

'Set return value

'Abort the transaction
CtxSetAbort

'Raise the error to the calling module
RaiseError TypeName(Me), "UpdManfProdLinePr(...)"

```

Exit Function

End Function

```

'=====
'Adds pricing information for a commodity associated with a manufacturer.
'2000/01/28 chrubecky added extra transaction complete before stored procedure
call.
'=====

```

```

Public Function CreateManfCommPr(ByVal alManfID As Long, ByVal alCommId As Long, _
    ByVal adbListPr As Double, ByVal adbSalePr As Double, ByVal adbwebDiscVal As
Double, _
    ByVal aswebDiscTypCd As String) As ADODB.Recordset
    On Error GoTo CreateManfCommPrErr

```

```

Dim oDB As DBCommPricing
Dim lRtn As Long
Dim sSQL As String

```

```

'If a discount type code was entered, validate it.
aswebDiscTypCd = Trim(aswebDiscTypCd)
If aswebDiscTypCd <> "" Then
    If aswebDiscTypCd <> CStr(DiscountTypeCode.ecDollar) And _
        aswebDiscTypCd <> CStr(DiscountTypeCode.ecPercentage) Then
        RaiseResError DawnProdErrCodes.ecInvalidDiscTypCd, TypeName(Me),

```

BProd.cls

```

"CreateManfCommPr(...)"
    ElseIf adbwebDiscVal < 0 Then
        RaiseResError DawnProdErrCodes.ecWebDiscountValueRequired, TypeName(Me),
    -
        "CreateManfCommPr(...)"
    End If
End If

'Allow the DB component to do the actual insert.
Set oDB = CtxCreateObject("DawnProd.DBCommPricing")
lRtn = oDB.CreateManfCommPr(alManfID, alCommId, adbListPr, adbsalePr,
adbwebDiscVal, _
    aswebDiscTypCd)
If lRtn <> 1 Then 'Something went wrong on the insert.
    RaiseResError DawnProdErrCodes.ecInsertFailed, TypeName(Me),
"CreateManfCommPr(...)"
Else
    'Now get the newly created record.
    sSQL = "SELECT mfr_id, commodity_id, list_price, sale_price, web_disc_val,
web_disc_typ_cd," & _
    last_maint_dtm FROM mfr_comm_pricing WHERE mfr_id = " &
CStr(alManfID) & _
    "AND commodity_id = " & CStr(alCommId)
    Set CreateManfCommPr = GetSQLRecordset(sSQL)
End If

'Tell MTS the transaction is complete.
CtxSetComplete

'Release object references
Set oDB = Nothing

Exit Function

CreateManfCommPrErr:
'Release object references
Set oDB = Nothing

'Set return value

'Abort the transaction
CtxSetAbort

'Raise the error to the calling module
RaiseError TypeName(Me), "CreateManfCommPr(...)"

Exit Function

End Function

'=====
'Updates the pricing information for a commodity associated with a manufacturer.
'2000/01/28 chrubecky added extra transaction complete before stored procedure
call.
'=====
Public Function UpdManfCommPr(ByVal alManfID As Long, ByVal alCommId As Long, _
    ByVal adbListPr As Double, ByVal adbsalePr As Double, ByVal adbwebDiscVal As
Double, _
    ByVal aswebDiscTypCd As String, ByVal adtLastMaintDtm As Date) As
ADODB.Recordset
    On Error GoTo UpdManfCommPrErr

```


BProd.cls

```

Dim oDB As DBCommPricing
Dim lRtn As Long
Dim sSQL As String
Dim oRS As ADODB.Recordset

'If a discount type code was entered, validate it.
asWebDiscTypCd = Trim(asWebDiscTypCd)
If asWebDiscTypCd <> "" Then
    If asWebDiscTypCd <> CStr(DiscountTypeCode.ecDollar) And _
        asWebDiscTypCd <> CStr(DiscountTypeCode.ecPercentage) Then
        RaiseResError DawnProdErrCodes.ecInvalidDiscTypCd, TypeName(Me),
"UpdManfCommPr(...)"
    ElseIf adbWebDiscVal < 0 Then
        RaiseResError DawnProdErrCodes.ecWebDiscountValueRequired, TypeName(Me),
-
        "UpdManfCommPr(...)"
    End If
End If

'Allow the DB component to do the actual update.
Set oDB = CtxCreateObject("DawnProd.DBCommPricing")
lRtn = oDB.UpdManfCommPr(alManfID, alCommId, adbListPr, adbSalePr,
adbWebDiscVal, _
    asWebDiscTypCd, adtLastMaintDtm)
If lRtn <> 1 Then 'Something went wrong on the update.
    sSQL = "SELECT Count(*) as rtn FROM mfr_comm_pricing WHERE mfr_id = " &
CStr(alManfID) & _
        " AND commodity_id = " & CStr(alCommId)
    Set oRS = GetSQLRecordset(sSQL)
    If Not oRS.EOF Then
        If oRS.Fields("rtn").Value = 0 Then
            RaiseResError DawnProdErrCodes.ecRecordDeleted, TypeName(Me),
"UpdManfCommPr(...)"
        Else 'Row changed between retrieve and update.
            RaiseResError DawnProdErrCodes.ecRecordChanged, TypeName(Me),
"UpdManfCommPr(...)"
        End If
    End If
Else
    'Now get the newly updated record.
    sSQL = "SELECT mfr_id, commodity_id, list_price, sale_price, web_disc_val,
web_disc_typ_cd," & _
        " last_maint_dtm FROM mfr_comm_pricing WHERE mfr_id = " &
CStr(alManfID) & _
        " AND commodity_id = " & CStr(alCommId)
    Set UpdManfCommPr = GetSQLRecordset(sSQL)
End If

'Tell MTS the transaction is complete.
CtxSetComplete

'Release object references
Set oDB = Nothing
Set oRS = Nothing

Exit Function

UpdManfCommPrErr:
'Release object references
Set oDB = Nothing
Set oRS = Nothing

```

```

'Set return value

'Abort the transaction
CtxSetAbort

'Raise the error to the calling module
RaiseError TypeName(Me), "UpdManfCommPr(...)"

Exit Function

End Function

'=====
'Adds pricing information for a part associated with a manufacturer.
'=====
Public Function CreateManfPartPr(ByVal alManfID As Long, ByVal alPartID As Long, _
    ByVal adbListPr As Double, ByVal adbSalePr As Double, ByVal adbwebDiscVal As
Double, _
    ByVal aswebDiscTypCd As String) As ADODB.Recordset
    On Error GoTo CreateManfPartPrErr

    Dim oDB As DBPartPricing
    Dim lRtn As Long
    Dim sSQL As String

    'If a discount type code was entered, validate it.
    aswebDiscTypCd = Trim(aswebDiscTypCd)
    If aswebDiscTypCd <> "" Then
        If aswebDiscTypCd <> CStr(DiscountTypeCode.ecDollar) And _
            aswebDiscTypCd <> CStr(DiscountTypeCode.ecPercentage) Then
            RaiseResError DawnProdErrCodes.ecInvalidDiscTypCd, TypeName(Me),
"CreateManfPartPr(...)"
        ElseIf adbwebDiscVal < 0 Then
            RaiseResError DawnProdErrCodes.ecWebDiscountValueRequired, TypeName(Me),
-
            "CreateManfPartPr(...)"
        End If
    End If

    'Allow the DB component to do the actual insert.
    Set oDB = CtxCreateObject("DawnProd.DBPartPricing")
    lRtn = oDB.CreateManfPartPr(alManfID, alPartID, adbListPr, adbSalePr,
adbwebDiscVal, _
        aswebDiscTypCd)
    If lRtn <> 1 Then 'Something went wrong on the insert.
        RaiseResError DawnProdErrCodes.ecInsertFailed, TypeName(Me),
"CreateManfPartPr(...)"
    Else
        'Now get the newly created record.
        sSQL = "SELECT mfr_id, part_id, list_price, sale_price, web_disc_val,
web_disc_typ_cd," & _
            " last_maint_dtm FROM mfr_part_pricing WHERE mfr_id = " &
CStr(alManfID) & _
            "AND part_id = " & CStr(alPartID)
        Set CreateManfPartPr = GetSQLRecordset(sSQL)
    End If

    'Tell MTS the transaction is complete.
    CtxSetComplete

```

```

'Release object references
Set oDB = Nothing

Exit Function

CreateManfPartPrErr:
'Release object references
Set oDB = Nothing

'Set return value

'Abort the transaction
CtxSetAbort

'Raise the error to the calling module
RaiseError TypeName(Me), "CreateManfPartPr(...)"

Exit Function

End Function

'=====
'Updates the pricing information for a part associated with a manufacturer.
'=====
Public Function UpdManfPartPr(ByVal alManfID As Long, ByVal alPartID As Long, _
    ByVal adbListPr As Double, ByVal adbsalePr As Double, ByVal adbwebDiscVal As
Double, _
    ByVal aswebDiscTypCd As String, ByVal adtLastMaintDtm As Date) As
ADODB.Recordset
    On Error GoTo UpdManfPartPrErr

    Dim oDB As DBPartPricing
    Dim lRtn As Long
    Dim sSQL As String
    Dim oRS As ADODB.Recordset

    'If a discount type code was entered, validate it.
    aswebDiscTypCd = Trim(aswebDiscTypCd)
    If aswebDiscTypCd <> "" Then
        If aswebDiscTypCd <> CStr(DiscountTypeCode.ecDollar) And _
            aswebDiscTypCd <> CStr(DiscountTypeCode.ecPercentage) Then
            RaiseResError DawnProdErrCodes.ecInvalidDiscTypCd, TypeName(Me),
"UpdManfPartPr(...)"
        ElseIf adbwebDiscVal < 0 Then
            RaiseResError DawnProdErrCodes.ecWebDiscountValueRequired, TypeName(Me),
-
            "UpdManfPartPr(...)"
        End If
    End If

    'Allow the DB component to do the actual update.
    Set oDB = CtxCreateObject("DawnProd.DBPartPricing")
    lRtn = oDB.UpdManfPartPr(alManfID, alPartID, adbListPr, adbsalePr,
adbwebDiscVal, _
        aswebDiscTypCd, adtLastMaintDtm)
    If lRtn <> 1 Then 'Something went wrong on the update.
        sSQL = "SELECT Count(*) as rtn FROM mfr_part_pricing WHERE mfr_id = " &
CStr(alManfID) & _
            " AND part_id = " & CStr(alPartID)
        Set oRS = GetSQLRecordset(sSQL)
        If Not oRS.EOF Then

```

```

        BProd.cls
        If oRS.Fields("rtn").Value = 0 Then
            RaiseResError DawnProdErrCodes.ecRecordDeleted, TypeName(Me),
"UpdManfPartPr(...)"
        Else 'Row changed between retrieve and update.
            RaiseResError DawnProdErrCodes.ecRecordChanged, TypeName(Me),
"UpdManfPartPr(...)"
        End If
    End If
Else
    'Now get the newly updated record.
    sSQL = "SELECT mfr_id, part_id, list_price, sale_price, web_disc_val,
web_disc_typ_cd," & _
        CStr(alManfID) & _
        " last_maint_dtm FROM mfr_part_pricing WHERE mfr_id = " &
        " AND part_id = " & CStr(alPartID)
    Set UpdManfPartPr = GetSQLRecordset(sSQL)
End If

'Tell MTS the transaction is complete.
CtxSetComplete

'Release object references
Set oDB = Nothing
Set oRS = Nothing

Exit Function

```

```

UpdManfPartPrErr:
'Release object references
Set oDB = Nothing
Set oRS = Nothing

'Set return value

'Abort the transaction
CtxSetAbort

'Raise the error to the calling module
RaiseError TypeName(Me), "UpdManfPartPr(...)"

Exit Function

End Function

```

```

'=====
'Adds a discount for a default product line for a manufacturer.
'2000/01/28 chrubecky added extra transaction complete before stored procedure
call.
'=====
Public Function CreateManfDefProdLineDisc(ByVal alManfID As Long, ByVal alProdLineID
As Long, _
    ByVal asDiscStusCd As String, ByVal asDiscTypCd As String, ByVal adbDiscVal
As Double, ByVal adDiscStartDte As Date, _
    ByVal adDiscEndDte As Date) As ADODB.Recordset
    On Error GoTo CreateManfDefProdLineDiscErr

    Dim lDiscId As Long
    Dim lRtn As Long
    Dim oBSeq As DawnSeq.BSequence
    Dim oDB As DBGblProdLineDisc

```

BProd.cls

```

Dim sSQL As String
Dim oRS As ADODB.Recordset

'Make sure the DiscTypCd and DiscStusCd are valid.
asDiscTypCd = Trim(asDiscTypCd)
asDiscStusCd = Trim(asDiscStusCd)
If asDiscTypCd <> CStr(DiscountTypeCode.ecDollar) And asDiscTypCd <> _
    CStr(DiscountTypeCode.ecPercentage) Then
    RaiseResError DawnProdErrCodes.ecInvalidDiscTypCd, TypeName(Me), _
        "CreateManfDefProdLineDisc(...)"
End If
If asDiscStusCd <> CStr(DiscountStatusCode.ecActive) And asDiscStusCd <> _
    CStr(DiscountStatusCode.ecInactive) Then
    RaiseResError DawnProdErrCodes.ecInvalidDiscStusCd, TypeName(Me), _
        "CreateManfDefProdLineDisc(...)"
End If

'make sure that if the discount has an end date, this does not clash with an
already active discount with an end date
'this check is for special discounts
If addDiscEndDte > 0 Then
    sSQL = "SELECT 1 FROM gbl_prod_line_disc WHERE disc_start_dte is not null
and disc_end_dte is not null "
    sSQL = sSQL & " AND mfr_id = " & CStr(alManfID) & " AND prod_line_id = " &
CStr(alProdLineID)
    sSQL = sSQL & " AND (" & GetSQLDate(adDiscStartDte) & " BETWEEN
disc_start_dte AND disc_end_dte "
    sSQL = sSQL & " OR " & GetSQLDate(adDiscEndDte) & " BETWEEN disc_start_dte
AND disc_end_dte "
    sSQL = sSQL & " OR disc_start_dte BETWEEN " & GetSQLDate(adDiscStartDte) &
" AND " & GetSQLDate(adDiscEndDte)
    sSQL = sSQL & " OR disc_end_dte BETWEEN " & GetSQLDate(adDiscStartDte) & "
AND " & GetSQLDate(adDiscEndDte) & " ) "
    sSQL = sSQL & " AND disc_stus_cd = '1' "
    Set oRS = GetSQLRecordset(sSQL)
    If oRS.RecordCount > 0 Then
        RaiseResError DawnProd.DawnProdErrCodes.ecDuplicateSpecial,
            TypeName(Me), "CreateManfDefProdLineDisc"
    End If
Else
    sSQL = " SELECT 1 FROM gbl_prod_line_disc WHERE disc_start_dte is NOT NULL
AND disc_end_dte IS NULL "
    sSQL = sSQL & " AND mfr_id = " & CStr(alManfID) & " AND prod_line_id = " &
CStr(alProdLineID)
    sSQL = sSQL & " AND disc_start_dte = " & GetSQLDate(adDiscStartDte)
    Set oRS = GetSQLRecordset(sSQL)
    If oRS.RecordCount > 0 Then
        RaiseResError DawnProd.DawnProdErrCodes.ecDuplicateDisc, TypeName(Me),
            "CreateManfDefProdLineDisc"
    End If
End If

'Get the next disc_id.
Set oBSeq = CtxCreateObject("DawnSeq.BSequence")
lDiscId = oBSeq.GetNextInSequence("discount")
If lDiscId < 1 Then
    RaiseResError DawnProdErrCodes.ecSequenceError, TypeName(Me),
        "CreateManfDefProdLineDisc(...)"
End If

'Allow the DB component to do the actual insert.
Set oDB = CtxCreateObject("DawnProd.DBGblProdLineDisc")
lRtn = oDB.CreateManfDefProdLineDisc(alManfID, alProdLineID, lDiscId,

```

BProd.cls

```

asDiscStusCd, _
    asDiscTypCd, adbDiscVal, addDiscStartDte, addDiscEndDte)

    If lRtn <> 1 Then 'Something went wrong with the insert
        RaiseResError DawnProdErrCodes.ecInsertFailed, TypeName(Me),
"CreateManfDefProdLineDisc(...)"
    Else
        'Retrieve the newly created record.
        sSQL = "SELECT mfr_id, prod_line_id, disc_id, disc_stus_cd, disc_typ_cd,
disc_val," & _
            "disc_start_dte, disc_end_dte, last_maint_dtm FROM gbl_prod_line_disc"
        & _
            " WHERE mfr_id = " & CStr(alManfID) & " AND prod_line_id = " &
CStr(alProdLineID) & _
            " AND disc_id = " & CStr(lDiscId)

        Set CreateManfDefProdLineDisc = GetSQLRecordset(sSQL)
    End If

    'Tell MTS we're done.
    CtxSetComplete

    'Release object references
    Set oDB = Nothing
    Set oBSeq = Nothing
    Set oRS = Nothing

    Exit Function

CreateManfDefProdLineDiscErr:
    'Release object references
    Set oDB = Nothing
    Set oBSeq = Nothing
    Set oRS = Nothing

    'Set return value

    'Abort the transaction
    CtxSetAbort

    'Raise the error to the calling module
    RaiseError TypeName(Me), "CreateManfDefProdLineDisc(...)"

    Exit Function

End Function

'=====
'Updates a discount for a default product line for a manufacturer.
'2000/01/28 chrubecky added extra transaction complete before stored procedure
call.
'=====
Public Function UpdManfDefProdLineDisc(ByVal alManfID As Long, ByVal alProdLineID As
Long, ByVal alDiscId As Long, _
    ByVal asDiscStusCd As String, ByVal asDiscTypCd As String, ByVal adbDiscVal
As Double, ByVal addDiscStartDte As Date, _
    ByVal addDiscEndDte As Date, ByVal adtLastMaintDtm As Date) As
ADODB.Recordset
    On Error GoTo UpdManfDefProdLineDiscErr

    Dim oDB As DBGblProdLineDisc

```

BProd.cls

```

Dim lRtn As Long
Dim sSQL As String
Dim oRS As ADODB.Recordset
Dim oRDDisc As ADODB.Recordset

'Make sure the DiscTypCd and DiscStusCd are valid.
asDiscTypCd = Trim(asDiscTypCd)
asDiscStusCd = Trim(asDiscStusCd)
If asDiscTypCd <> CStr(DiscountTypeCode.ecDollar) And asDiscTypCd <> _
    CStr(DiscountTypeCode.ecPercentage) Then
    RaiseResError DawnProdErrCodes.ecInvalidDiscTypCd, TypeName(Me), _
        "UpdManfDefProdLineDisc(...)"
End If
If asDiscStusCd <> CStr(DiscountStatusCode.ecActive) And asDiscStusCd <> _
    CStr(DiscountStatusCode.ecInactive) Then
    RaiseResError DawnProdErrCodes.ecInvalidDiscStusCd, TypeName(Me), _
        "UpdManfDefProdLineDisc(...)"
End If

'make sure that if the discount has an end date, this does not clash with an
already active discount with an end date
'this check is for special discounts
If asDiscStusCd = ecActive Then
    If addDiscEndDte > 0 Then
        sSQL = "SELECT 1 FROM gbl_prod_line_disc WHERE disc_start_dte is not
null and disc_end_dte is not null
        sSQL = sSQL & " AND mfr_id = " & CStr(alManfID) & " AND prod_line_id = "
& CStr(alProdLineID)
        sSQL = sSQL & " AND disc_id <> " & CStr(alDiscId)
        sSQL = sSQL & " AND (" & GetSQLDate(addDiscStartDte) & " BETWEEN
disc_start_dte AND disc_end_dte "
        sSQL = sSQL & " OR " & GetSQLDate(addDiscEndDte) & " BETWEEN
disc_start_dte AND disc_end_dte "
        sSQL = sSQL & " OR disc_start_dte BETWEEN " &
GetSQLDate(addDiscStartDte) & " AND " & GetSQLDate(addDiscEndDte)
        sSQL = sSQL & " OR disc_end_dte BETWEEN " & GetSQLDate(addDiscStartDte)
& " AND " & GetSQLDate(addDiscEndDte) & " ) "
        sSQL = sSQL & " AND disc_stus_cd = '1' "
        Set oRDDisc = GetSQLRecordset(sSQL)
        If oRDDisc.RecordCount > 0 Then
            RaiseResError DawnProd.DawnProdErrCodes.ecDuplicateSpecial,
                TypeName(Me), "UpdManfDefProdLineDisc"
        End If
    Else
        sSQL = " SELECT 1 FROM gbl_prod_line_disc WHERE disc_start_dte is NOT
NULL AND disc_end_dte IS NULL "
        sSQL = sSQL & " AND mfr_id = " & CStr(alManfID) & " AND prod_line_id = "
& CStr(alProdLineID)
        sSQL = sSQL & " AND disc_id <> " & CStr(alDiscId)
        sSQL = sSQL & " AND disc_start_dte = " & GetSQLDate(addDiscStartDte)
        Set oRS = GetSQLRecordset(sSQL)
        If oRS.RecordCount > 0 Then
            RaiseResError DawnProd.DawnProdErrCodes.ecDuplicateDisc,
                TypeName(Me), "UpdManfDefProdLineDisc"
        End If
    End If
End If

'Allow the DB component to do the actual update.
Set oDB = CtxCreateObject("DawnProd.DBGblProdLineDisc")
lRtn = oDB.UpdManfDefProdLineDisc(alManfID, alProdLineID, alDiscId,
asDiscStusCd, asDiscTypCd, _
    adbDiscVal, addDiscStartDte, addDiscEndDte, adtLastMaintDtm)

```

```

                                BProd.cls
If lRtn <> 1 Then 'Something went wrong on the update.
    sSQL = "SELECT Count(*) as rtn FROM gbl_prod_line_disc" & _
        " WHERE mfr_id = " & CStr(alManfID) & " AND prod_line_id = " &
CStr(alProdLineID) & _
        " AND disc_id = " & CStr(alDiscID)
    Set oRS = GetSQLRecordset(sSQL)
    If Not oRS.EOF Then
        If oRS.Fields("rtn").Value = 0 Then
            RaiseResError DawnProdErrCodes.ecRecordDeleted, TypeName(Me),
"UpdManfDefProdLineDisc(...)"
        Else 'Row changed between retrieve and update.
            RaiseResError DawnProdErrCodes.ecRecordChanged, TypeName(Me),
"UpdManfDefProdLineDisc(...)"
        End If
    End If
Else
    'Now get the newly updated record.
    sSQL = "SELECT mfr_id, prod_line_id, disc_id, disc_stus_cd, disc_typ_cd,
disc_val," & _
        " disc_start_dte, disc_end_dte, last_maint_dtm FROM gbl_prod_line_disc"
    & _
        " WHERE mfr_id = " & CStr(alManfID) & " AND prod_line_id = " &
CStr(alProdLineID) & _
        " AND disc_id = " & CStr(alDiscID)
    Set UpdManfDefProdLineDisc = GetSQLRecordset(sSQL)
End If

'Tell MTS the transaction is complete.
CtxSetComplete

'Release object references
Set oDB = Nothing
Set oRS = Nothing
Set oRDDisc = Nothing

Exit Function

UpdManfDefProdLineDiscErr:
'Release object references
Set oDB = Nothing
Set oRS = Nothing
Set oRDDisc = Nothing

'Set return value

'Abort the transaction
CtxSetAbort

'Raise the error to the calling module
RaiseError TypeName(Me), "UpdManfDefProdLineDisc(...)"

Exit Function

End Function

```

```

'=====
'Deletes a discount for a default product line for a manufacturer.
'2000/01/28 chrubecky added extra transaction complete before stored procedure
call.
'=====

```

```

Public Function DelManfDefProdLineDisc(ByVal alManfID As Long, ByVal alProdLineID As
Page 37

```


BProd.cls

```

Long, ByVal alDiscId As Long, _
    ByVal adtLastMaintDtm As Date)
    On Error GoTo DelManfDefProdLineDiscErr

    Dim oDB As DBGblProdLineDisc
    Dim lRtn As Long
    Dim sSQL As String
    Dim oRS As ADODB.Recordset

    'Allow the DB component to do the actual deletion.
    Set oDB = CtxCreateObject("DawnProd.DBGblProdLineDisc")
    lRtn = oDB.DelManfDefProdLineDisc(alManfID, alProdLineID, alDiscId,
adtLastMaintDtm)
    If lRtn <> 1 Then 'something went wrong
        sSQL = "SELECT Count(*) as rtn FROM gbl_prod_line_disc" & _
            " WHERE mfr_id = " & CStr(alManfID) & " AND prod_line_id = " &
CStr(alProdLineID) & _
            " AND disc_id = " & CStr(alDiscId)
        Set oRS = GetSQLRecordset(sSQL)
        If Not oRS.EOF Then
            If oRS.Fields("rtn").Value = 0 Then
                RaiseResError DawnProdErrCodes.ecRecordAlreadyDeleted, TypeName(Me),
"DelManfDefProdLineDisc(...)"
            Else 'Row changed between retrieve and delete.
                RaiseResError DawnProdErrCodes.ecRecordChanged, TypeName(Me),
"DelManfDefProdLineDisc(...)"
            End If
        End If
    End If

    'Tell MTS the transaction is complete.
    CtxSetComplete

    'Release object references
    Set oDB = Nothing
    Set oRS = Nothing

    Exit Function

DelManfDefProdLineDiscErr:
    'Release object references
    Set oDB = Nothing
    Set oRS = Nothing

    'Set return value

    'Abort the transaction
    CtxSetAbort

    'Raise the error to the calling module
    RaiseError TypeName(Me), "DelManfDefProdLineDisc(...)"

    Exit Function

End Function

'=====
'Adds a discount for a default commodity for a manufacturer.
'2000/01/28 chrubecky added extra transaction complete before stored procedure
call.
'=====

```

```

                                BProd.cls
Public Function CreateManfDefCommDisc(ByVal alManfID As Long, ByVal alCommId As
Long, _
    ByVal asDiscStusCd As String, ByVal asDiscTypCd As String, ByVal adbDiscVal
As Double, ByVal addDiscStartDte As Date, _
    ByVal addDiscEndDte As Date) As ADODB.Recordset
    On Error GoTo CreateManfDefCommDiscErr

    Dim lDiscId As Long
    Dim lRtn As Long
    Dim oBSeq As DawnSeq.BSequence
    Dim oDB As DBGblCommDisc
    Dim sSQL As String
    Dim oRS As ADODB.Recordset

    'Make sure the DiscTypCd and DiscStusCd are valid.
    asDiscTypCd = Trim(asDiscTypCd)
    asDiscStusCd = Trim(asDiscStusCd)
    If asDiscTypCd <> CStr(DiscountTypeCode.ecDollar) And asDiscTypCd <> _
        CStr(DiscountTypeCode.ecPercentage) Then
        RaiseResError DawnProdErrCodes.ecInvalidDiscTypCd, TypeName(Me), _
            "CreateManfDefCommDisc(...)"
    End If
    If asDiscStusCd <> CStr(DiscountStatusCode.ecActive) And asDiscStusCd <> _
        CStr(DiscountStatusCode.ecInactive) Then
        RaiseResError DawnProdErrCodes.ecInvalidDiscStusCd, TypeName(Me), _
            "CreateManfDefCommDisc(...)"
    End If

    'make sure that if the discount has an end date, this does not clash with an
    already active discount with an end date
    'this check is for special discounts
    If addDiscEndDte > 0 Then
        sSQL = "SELECT 1 FROM gbl_comm_disc WHERE disc_start_dte is not null and
disc_end_dte is not null "
        sSQL = sSQL & " AND mfr_id = " & CStr(alManfID) & " AND commodity_id = " &
CStr(alCommId)
        sSQL = sSQL & " AND (" & GetSQLDate(addDiscStartDte) & " BETWEEN
disc_start_dte AND disc_end_dte "
        sSQL = sSQL & " OR " & GetSQLDate(addDiscEndDte) & " BETWEEN disc_start_dte
AND disc_end_dte "
        sSQL = sSQL & " OR disc_start_dte BETWEEN " & GetSQLDate(addDiscStartDte) &
" AND " & GetSQLDate(addDiscEndDte)
        sSQL = sSQL & " OR disc_end_dte BETWEEN " & GetSQLDate(addDiscStartDte) & "
AND " & GetSQLDate(addDiscEndDte) & ") "
        sSQL = sSQL & " AND disc_stus_cd = '1' "
        Set oRS = GetSQLRecordset(sSQL)
        If oRS.RecordCount > 0 Then
            RaiseResError DawnProd.DawnProdErrCodes.ecDuplicateSpecial,
                TypeName(Me), "CreateManfDefCommDisc"
        End If
    Else
        sSQL = " SELECT 1 FROM gbl_comm_disc WHERE disc_start_dte is NOT NULL AND
disc_end_dte IS NULL "
        sSQL = sSQL & " AND mfr_id = " & CStr(alManfID) & " AND commodity_id = " &
CStr(alCommId)
        sSQL = sSQL & " AND disc_start_dte = " & GetSQLDate(addDiscStartDte)
        Set oRS = GetSQLRecordset(sSQL)
        If oRS.RecordCount > 0 Then
            RaiseResError DawnProd.DawnProdErrCodes.ecDuplicateDisc, TypeName(Me),
                "CreateManfDefCommDisc"
        End If
    End If
End If

```

BProd.cls

```
'Get the next disc_id.
Set oBSeq = CtxCreateObject("DawnSeq.BSequence")
lDiscId = oBSeq.GetNextInSequence("discount")
If lDiscId < 1 Then
    RaiseResError DawnProdErrCodes.ecSequenceError, TypeName(Me),
"CreateManfDefCommDisc(...)"
End If

'Allow the DB component to do the actual insert.
Set oDB = CtxCreateObject("DawnProd.DBGblCommDisc")
lRtn = oDB.CreateManfDefCommDisc(alManfID, alCommId, lDiscId, asDiscStusCd, _
    asDiscTypCd, adbDiscVal, addDiscStartDte, addDiscEndDte)

If lRtn <> 1 Then 'Something went wrong with the insert
    RaiseResError DawnProdErrCodes.ecInsertFailed, TypeName(Me),
"CreateManfDefCommDisc(...)"
Else
    'Retrieve the newly created record.
    sSQL = "SELECT mfr_id, commodity_id, disc_id, disc_stus_cd, disc_typ_cd,
disc_val," & _
        " disc_start_dte, disc_end_dte, last_maint_dtm FROM gbl_comm_disc" & _
        " WHERE mfr_id = " & CStr(alManfID) & " AND commodity_id = " & _
CStr(alCommId) & _
        " AND disc_id = " & CStr(lDiscId)

    Set CreateManfDefCommDisc = GetSQLRecordset(sSQL)
End If

'Tell MTS we're done.
CtxSetComplete

'Release object references
Set oDB = Nothing
Set oBSeq = Nothing
Set oRS = Nothing
Exit Function

CreateManfDefCommDiscErr:
'Release object references
Set oDB = Nothing
Set oBSeq = Nothing
Set oRS = Nothing

'Set return value

'Abort the transaction
CtxSetAbort

'Raise the error to the calling module
RaiseError TypeName(Me), "CreateManfDefCommDisc(...)"

Exit Function

End Function
```

```
'=====
'Updates a discount for a default commodity for a manufacturer.
'2000/01/28 chrubecky added extra transaction complete before stored procedure
call.
'=====
```

```
Public Function UpdManfDefCommDisc(ByVal alManfID As Long, ByVal alCommId As Long,
Page 40
```

BProd.cls

```

ByVal alDiscId As Long, _
    ByVal asDiscStusCd As String, ByVal asDiscTypCd As String, ByVal adbDiscVal
As Double, ByVal addDiscStartDte As Date, _
    ByVal addDiscEndDte As Date, ByVal adtLastMaintDtm As Date) As
ADODB.Recordset
    On Error GoTo UpdManfDefCommDiscErr

    Dim oDB As DBGblCommDisc
    Dim lRtn As Long
    Dim sSQL As String
    Dim oRS As ADODB.Recordset
    Dim oRSDisc As ADODB.Recordset

    'Make sure the DiscTypCd and DiscStusCd are valid.
    asDiscTypCd = Trim(asDiscTypCd)
    asDiscStusCd = Trim(asDiscStusCd)
    If asDiscTypCd <> CStr(DiscountTypeCode.ecDollar) And asDiscTypCd <> _
        CStr(DiscountTypeCode.ecPercentage) Then
        RaiseResError DawnProdErrCodes.ecInvalidDiscTypCd, TypeName(Me), _
            "UpdManfDefCommDisc(...)"
    End If
    If asDiscStusCd <> CStr(DiscountStatusCode.ecActive) And asDiscStusCd <> _
        CStr(DiscountStatusCode.ecInactive) Then
        RaiseResError DawnProdErrCodes.ecInvalidDiscStusCd, TypeName(Me), _
            "UpdManfDefCommDisc(...)"
    End If

    'make sure that if the discount has an end date, this does not clash with an
    already active discount with an end date
    'this check is for special discounts
    If asDiscStusCd = ecActive Then
        If addDiscEndDte > 0 Then
            sSQL = "SELECT 1 FROM gbl_comm_disc WHERE disc_start_dte is not null and
disc_end_dte is not null "
            sSQL = sSQL & " AND mfr_id = " & CStr(alManfID) & " AND commodity_id = "
& CStr(alCommId)
            sSQL = sSQL & " AND disc_id <> " & CStr(alDiscId)
            sSQL = sSQL & " AND (" & GetSQLDate(addDiscStartDte) & " BETWEEN
disc_start_dte AND disc_end_dte "
            sSQL = sSQL & " OR " & GetSQLDate(addDiscEndDte) & " BETWEEN
disc_start_dte AND disc_end_dte "
            sSQL = sSQL & " OR disc_start_dte BETWEEN " &
GetSQLDate(addDiscStartDte) & " AND " & GetSQLDate(addDiscEndDte)
            sSQL = sSQL & " OR disc_end_dte BETWEEN " & GetSQLDate(addDiscStartDte)
& " AND " & GetSQLDate(addDiscEndDte) & " ) "
            sSQL = sSQL & " AND disc_stus_cd = '1' "
            Set oRSDisc = GetSQLRecordset(sSQL)
            If oRSDisc.RecordCount > 0 Then
                RaiseResError DawnProd.DawnProdErrCodes.ecDuplicateSpecial,
TypeName(Me), "UpdManfDefCommDisc"
            End If
        Else
            sSQL = " SELECT 1 FROM gbl_comm_disc WHERE disc_start_dte is NOT NULL
AND disc_end_dte IS NULL "
            sSQL = sSQL & " AND mfr_id = " & CStr(alManfID) & " AND commodity_id = "
& CStr(alCommId)
            sSQL = sSQL & " AND disc_id <> " & CStr(alDiscId)
            sSQL = sSQL & " AND disc_start_dte = " & GetSQLDate(addDiscStartDte)
            Set oRS = GetSQLRecordset(sSQL)
            If oRS.RecordCount > 0 Then
                RaiseResError DawnProd.DawnProdErrCodes.ecDuplicateDisc,
TypeName(Me), "UpdManfDefCommDisc"
            End If
        End If
    End If

```

```

    End If
End If

'Allow the DB component to do the actual update.
Set oDB = CtxCreateObject("DawnProd.DBGblCommDisc")
lRtn = oDB.UpdManfDefCommDisc(alManfID, alCommId, alDiscId, asDiscStusCd,
asDiscTypCd, _
    adbDiscVal, addDiscStartDte, addDiscEndDte, addLastMaintDtm)
If lRtn <> 1 Then 'Something went wrong on the update.
    sSQL = "SELECT Count(*) as rtn FROM gbl_comm_disc" & _
        " WHERE mfr_id = " & CStr(alManfID) & " AND commodity_id = " &
CStr(alCommId) & _
        " AND disc_id = " & CStr(alDiscId)
    Set ORS = GetSQLRecordset(sSQL)
    If Not ORS.EOF Then
        If ORS.Fields("rtn").value = 0 Then
            RaiseResError DawnProdErrCodes.ecRecordDeleted, TypeName(Me),
"UpdManfDefCommDisc(...)"
        Else 'Row changed between retrieve and update.
            RaiseResError DawnProdErrCodes.ecRecordChanged, TypeName(Me),
"UpdManfDefCommDisc(...)"
        End If
    End If
Else
    'Now get the newly updated record.
    sSQL = "SELECT mfr_id, commodity_id, disc_id, disc_stus_cd, disc_typ_cd,
disc_val," & _
        " disc_start_dte, disc_end_dte, last_maint_dtm FROM gbl_comm_disc" & _
        " WHERE mfr_id = " & CStr(alManfID) & " AND commodity_id = " &
CStr(alCommId) & _
        " AND disc_id = " & CStr(alDiscId)
    Set UpdManfDefCommDisc = GetSQLRecordset(sSQL)
End If

'Tell MTS the transaction is complete.
CtxSetComplete

'Release object references
Set oDB = Nothing
Set ORS = Nothing
Set ORSDisc = Nothing

Exit Function

UpdManfDefCommDiscErr:
'Release object references
Set oDB = Nothing
Set ORS = Nothing
Set ORSDisc = Nothing

'Set return value

'Abort the transaction
CtxSetAbort

'Raise the error to the calling module
RaiseError TypeName(Me), "UpdManfDefCommDisc(...)"

Exit Function

End Function

```

```

'=====

```

BProd.cls

```

=====
'Deletes a discount for a default commodity for a manufacturer.
'2000/01/28 chrubecky added extra transaction complete before stored procedure
call.
'=====
=====
Public Function DelManfDefCommDisc(ByVal alManfID As Long, ByVal alCommId As Long,
ByVal alDiscId As Long, _
ByVal adtLastMaintDtm As Date)
On Error GoTo DelManfDefCommDiscErr

Dim oDB As DBGblCommDisc
Dim lRtn As Long
Dim sSQL As String
Dim oRS As ADODB.Recordset

'Allow the DB component to do the actual deletion.
Set oDB = CtxCreateObject("DawnProd.DBGblCommDisc")
lRtn = oDB.DelManfDefCommDisc(alManfID, alCommId, alDiscId, adtLastMaintDtm)
If lRtn <> 1 Then 'something went wrong
    sSQL = "SELECT Count(*) as rtn FROM gbl_comm_disc" & _
        " WHERE mfr_id = " & CStr(alManfID) & " AND commodity_id = " &
CStr(alCommId) & _
        " AND disc_id = " & CStr(alDiscId)
    Set oRS = GetSQLRecordset(sSQL)
    If Not oRS.EOF Then
        If oRS.Fields("rtn").Value = 0 Then
            RaiseResError DawnProdErrCodes.ecRecordAlreadyDeleted, TypeName(Me),
"DelManfDefCommDisc(...)"
        Else 'Row changed between retrieve and delete.
            RaiseResError DawnProdErrCodes.ecRecordChanged, TypeName(Me),
"DelManfDefCommDisc(...)"
        End If
    End If
End If

'Tell MTS the transaction is complete.
CtxSetComplete

'Release object references
Set oDB = Nothing
Set oRS = Nothing

Exit Function

DelManfDefCommDiscErr:
'Release object references
Set oDB = Nothing
Set oRS = Nothing

'Set return value

'Abort the transaction
CtxSetAbort

'Raise the error to the calling module
RaiseError TypeName(Me), "DelManfDefCommDisc(...)"

Exit Function

End Function
=====

```

```

=====
'Adds a discount for a default part for a manufacturer.
'=====
=====
Public Function CreateManfDefPartDisc(ByVal alManfID As Long, ByVal alPartID As
Long, _
    ByVal asDiscStusCd As String, ByVal asDiscTypCd As String, ByVal adbDiscVal
As Double, ByVal addDiscStartDte As Date, _
    ByVal addDiscEndDte As Date) As ADODB.Recordset
    On Error GoTo CreateManfDefPartDiscErr

    Dim lDiscID As Long
    Dim lRtn As Long
    Dim oBSeq As DawnSeq.BSequence
    Dim oDB As DBGblPartDisc
    Dim sSQL As String
    Dim oRS As ADODB.Recordset

    'Make sure the DiscTypCd and DiscStusCd are valid.
    asDiscTypCd = Trim(asDiscTypCd)
    asDiscStusCd = Trim(asDiscStusCd)
    If asDiscTypCd <> CStr(DiscountTypeCode.ecDollar) And asDiscTypCd <> _
        CStr(DiscountTypeCode.ecPercentage) Then
        RaiseResError DawnProdErrCodes.ecInvalidDiscTypCd, TypeName(Me), _
            "CreateManfDefPartDisc(...)"
    End If
    If asDiscStusCd <> CStr(DiscountStatusCode.ecActive) And asDiscStusCd <> _
        CStr(DiscountStatusCode.ecInactive) Then
        RaiseResError DawnProdErrCodes.ecInvalidDiscStusCd, TypeName(Me), _
            "CreateManfDefPartDisc(...)"
    End If

    'make sure that if the discount has an end date, this does not clash with an
    already active discount with an end date
    'this check is for special discounts
    If addDiscEndDte > 0 Then
        sSQL = "SELECT 1 FROM gbl_part_disc WHERE disc_start_dte is not null and
disc_end_dte is not null "
        sSQL = sSQL & " AND mfr_id = " & CStr(alManfID) & " AND part_id = " &
CStr(alPartID)
        sSQL = sSQL & " AND (" & GetSQLDate(addDiscStartDte) & " BETWEEN
disc_start_dte AND disc_end_dte "
        sSQL = sSQL & " OR " & GetSQLDate(addDiscEndDte) & " BETWEEN disc_start_dte
AND disc_end_dte "
        sSQL = sSQL & " OR disc_start_dte BETWEEN " & GetSQLDate(addDiscStartDte) &
" AND " & GetSQLDate(addDiscEndDte)
        sSQL = sSQL & " OR disc_end_dte BETWEEN " & GetSQLDate(addDiscStartDte) & "
AND " & GetSQLDate(addDiscEndDte) & ") "
        sSQL = sSQL & " AND disc_stus_cd = '1' "
        Set oRS = GetSQLRecordset(sSQL)
        If oRS.RecordCount > 0 Then
            RaiseResError DawnProd.DawnProdErrCodes.ecDuplicateSpecial,
                TypeName(Me), "CreateManfDefPartDisc"
        End If
    Else
        sSQL = " SELECT 1 FROM gbl_part_disc WHERE disc_start_dte is NOT NULL AND
disc_end_dte IS NULL "
        sSQL = sSQL & " AND mfr_id = " & CStr(alManfID) & " AND part_id = " &
CStr(alPartID)
        sSQL = sSQL & " AND disc_start_dte = " & GetSQLDate(addDiscStartDte)
        Set oRS = GetSQLRecordset(sSQL)
        If oRS.RecordCount > 0 Then
            RaiseResError DawnProd.DawnProdErrCodes.ecDuplicateDisc, TypeName(Me),
                "CreateManfDefPartDisc"
        End If
    End If
End Function
=====

```

```

"CreateManfDefPartDisc"
    End If
End If

'Get the next disc_id.
Set oBSeq = CtxCreateObject("DawnSeq.BSequence")
lDiscId = oBSeq.GetNextInSequence("discount")
If lDiscId < 1 Then
    RaiseResError DawnProdErrCodes.ecSequenceError, TypeName(Me),
"CreateManfDefPartDisc(...)"
End If

'Allow the DB component to do the actual insert.
Set oDB = CtxCreateObject("DawnProd.DBGblPartDisc")
lRtn = oDB.CreateManfDefPartDisc(alManfID, alPartID, lDiscId, asDiscStusCd, _
    asDiscTypCd, adbDiscVal, addDiscStartDte, addDiscEndDte)

If lRtn <> 1 Then 'Something went wrong with the insert
    RaiseResError DawnProdErrCodes.ecInsertFailed, TypeName(Me),
"CreateManfDefPartDisc(...)"
Else
    'Retrieve the newly created record.
    SSQl = "SELECT mfr_id, part_id, disc_id, disc_stus_cd, disc_typ_cd,
disc_val," & _
        "disc_start_dte, disc_end_dte, last_maint_dtm FROM gbl_part_disc" & _
        " WHERE mfr_id = " & CStr(alManfID) & " AND part_id = " & CStr(alPartID)
    & _
        " AND disc_id = " & CStr(lDiscId)

    Set CreateManfDefPartDisc = GetSQLRecordset(SSQl)
End If

'Tell MTS we're done.
CtxSetComplete

'Release object references
Set oDB = Nothing
Set oBSeq = Nothing
Set oRS = Nothing

Exit Function

CreateManfDefPartDiscErr:
'Release object references
Set oDB = Nothing
Set oBSeq = Nothing
Set oRS = Nothing

'Set return value

'Abort the transaction
CtxSetAbort

'Raise the error to the calling module
RaiseError TypeName(Me), "CreateManfDefPartDisc(...)"

Exit Function

End Function

'=====
'Updates a discount for a default part for a manufacturer.

```



```

'=====
Public Function UpdManfDefPartDisc(ByVal alManfID As Long, ByVal alPartID As Long,
ByVal alDiscID As Long, _
    ByVal asDiscStusCd As String, ByVal asDiscTypCd As String, ByVal adbDiscVal
As Double, ByVal addDiscStartDte As Date, _
    ByVal addDiscEndDte As Date, ByVal adtLastMaintDtm As Date) As
ADODB.Recordset
    On Error GoTo UpdManfDefPartDiscErr

    Dim oDB As DBGblPartDisc
    Dim lRtn As Long
    Dim sSQL As String
    Dim oRS As ADODB.Recordset
    Dim oRSDisc As ADODB.Recordset

    'Make sure the DiscTypCd and DiscStusCd are valid.
    asDiscTypCd = Trim(asDiscTypCd)
    asDiscStusCd = Trim(asDiscStusCd)
    If asDiscTypCd <> CStr(DiscountTypeCode.ecDollar) And asDiscTypCd <> _
        CStr(DiscountTypeCode.ecPercentage) Then
        RaiseResError DawnProdErrCodes.ecInvalidDiscTypCd, TypeName(Me), _
            "UpdManfDefPartDisc(...)"
    End If
    If asDiscStusCd <> CStr(DiscountStatusCode.ecActive) And asDiscStusCd <> _
        CStr(DiscountStatusCode.ecInactive) Then
        RaiseResError DawnProdErrCodes.ecInvalidDiscStusCd, TypeName(Me), _
            "UpdManfDefPartDisc(...)"
    End If

    'make sure that if the discount has an end date, this does not clash with an
    already active discount with an end date
    'this check is for special discounts
    If asDiscStusCd = ecActive Then
        If addDiscEndDte > 0 Then
            sSQL = "SELECT 1 FROM gbl_part_disc WHERE disc_start_dte is not null and
disc_end_dte is not null "
            sSQL = sSQL & " AND mfr_id = " & CStr(alManfID) & " AND part_id = " &
CStr(alPartID)
            sSQL = sSQL & " AND disc_id <> " & CStr(alDiscID)
            sSQL = sSQL & " AND (" & GetSQLDate(addDiscStartDte) & " BETWEEN
disc_start_dte AND disc_end_dte "
            sSQL = sSQL & " OR " & GetSQLDate(addDiscEndDte) & " BETWEEN
disc_start_dte AND disc_end_dte "
            sSQL = sSQL & " OR disc_start_dte BETWEEN " &
GetSQLDate(addDiscStartDte) & " AND " & GetSQLDate(addDiscEndDte)
            sSQL = sSQL & " OR disc_end_dte BETWEEN " & GetSQLDate(addDiscStartDte)
& " AND " & GetSQLDate(addDiscEndDte) & " )"
            sSQL = sSQL & " AND disc_stus_cd = '1' "
            Set oRSDisc = GetSQLRecordset(sSQL)
            If oRSDisc.RecordCount > 0 Then
                RaiseResError DawnProd.DawnProdErrCodes.ecDuplicateSpecial,
TypeName(Me), "UpdManfDefPartDisc"
            End If
        Else
            sSQL = " SELECT 1 FROM gbl_part_disc WHERE disc_start_dte is NOT NULL
AND disc_end_dte IS NULL "
            sSQL = sSQL & " AND mfr_id = " & CStr(alManfID) & " AND part_id = " &
CStr(alPartID)
            sSQL = sSQL & " AND disc_id <> " & CStr(alDiscID)
            sSQL = sSQL & " AND disc_start_dte = " & GetSQLDate(addDiscStartDte)
            Set oRS = GetSQLRecordset(sSQL)
            If oRS.RecordCount > 0 Then

```

```

                                BProd.cls
    RaiseResError DawnProd.DawnProdErrCodes.ecDuplicateDisc,
    TypeName(Me), "UpdManfDefPartDisc"
    End If
    End If
    End If

    'Allow the DB component to do the actual update.
    Set oDB = CtxCreateObject("DawnProd.DBGblPartDisc")
    lRtn = oDB.UpdManfDefPartDisc(alManfID, alPartID, alDiscId, asDiscStusCd,
asDiscTypCd, _
    addbDiscVal, addDiscStartDte, addDiscEndDte, adtLastMaintDtm)
    If lRtn <> 1 Then 'Something went wrong on the update.
        sSQL = "SELECT Count(*) as rtn FROM gbl_part_disc" & _
        " WHERE mfr_id = " & CStr(alManfID) & " AND part_id = " & CStr(alPartID)
    & _
        " AND disc_id = " & CStr(alDiscId)
        Set oRS = GetSQLRecordset(sSQL)
        If Not oRS.EOF Then
            If oRS.Fields("rtn").Value = 0 Then
                RaiseResError DawnProdErrCodes.ecRecordDeleted, TypeName(Me),
                "UpdManfDefPartDisc(...)"
            Else 'Row changed between retrieve and update.
                RaiseResError DawnProdErrCodes.ecRecordChanged, TypeName(Me),
                "UpdManfDefPartDisc(...)"
            End If
        End If
    Else
        'Now get the newly updated record.
        sSQL = "SELECT mfr_id, part_id, disc_id, disc_stus_cd, disc_typ_cd,
disc_val," & _
        " disc_start_dte, disc_end_dte, last_maint_dtm FROM gbl_part_disc" & _
        " WHERE mfr_id = " & CStr(alManfID) & " AND part_id = " & CStr(alPartID)
    & _
        " AND disc_id = " & CStr(alDiscId)
        Set UpdManfDefPartDisc = GetSQLRecordset(sSQL)
    End If

    'Tell MTS the transaction is complete.
    CtxSetComplete

    'Release object references
    Set oDB = Nothing
    Set oRS = Nothing
    Set ORSDisc = Nothing

    Exit Function

UpdManfDefPartDiscErr:
    'Release object references
    Set oDB = Nothing
    Set oRS = Nothing
    Set ORSDisc = Nothing

    'Set return value

    'Abort the transaction
    CtxSetAbort

    'Raise the error to the calling module
    RaiseError TypeName(Me), "UpdManfDefPartDisc(...)"

    Exit Function

```

End Function

```

'=====
'Deletes a discount for a default part for a manufacturer.
'=====

```

```

Public Function DelManfDefPartDisc(ByVal alManfID As Long, ByVal alPartID As Long,
ByVal alDiscID As Long, _
    ByVal adtLastMaintDtm As Date)
    On Error GoTo DelManfDefPartDiscErr

    Dim oDB As DBGblPartDisc
    Dim lRtn As Long
    Dim sSQL As String
    Dim oRS As ADODB.Recordset

    'Allow the DB component to do the actual deletion.
    Set oDB = CtxCreateObject("DawnProd.DBGblPartDisc")
    lRtn = oDB.DelManfDefPartDisc(alManfID, alPartID, alDiscID, adtLastMaintDtm)
    If lRtn <> 1 Then 'something went wrong
        sSQL = "SELECT Count(*) as rtn FROM gbl_part_disc" & _
            " WHERE mfr_id = " & CStr(alManfID) & " AND part_id = " & CStr(alPartID)
        & _
            " AND disc_id = " & CStr(alDiscID)
        Set oRS = GetSQLRecordset(sSQL)
        If Not oRS.EOF Then
            If oRS.Fields("rtn").Value = 0 Then
                RaiseResError DawnProdErrCodes.ecRecordAlreadyDeleted, TypeName(Me),
"DelManfDefPartDisc(...)"
            Else 'Row changed between retrieve and delete.
                RaiseResError DawnProdErrCodes.ecRecordChanged, TypeName(Me),
"DelManfDefPartDisc(...)"
            End If
        End If
    End If

    'Tell MTS the transaction is complete.
    CtxSetComplete

    'Release object references
    Set oDB = Nothing
    Set oRS = Nothing

    Exit Function

```

```

DelManfDefPartDiscErr:
    'Release object references
    Set oDB = Nothing
    Set oRS = Nothing

    'Set return value

    'Abort the transaction
    CtxSetAbort

    'Raise the error to the calling module
    RaiseError TypeName(Me), "DelManfDefPartDisc(...)"

    Exit Function

```

End Function

BProd.cls

```

'=====
'Adds a discount for a product line for an organization.
'2000/01/28 chrubecky added extra transaction complete before stored procedure
call.
'=====
Public Function CreateOrgProdLineDisc(ByVal aOrgID As Long, ByVal aProdLineID As
Long, _
    ByVal asDiscStusCd As String, ByVal asDiscTypCd As String, ByVal adbDiscVal
As Double, ByVal addDiscStartDte As Date, _
    ByVal addDiscEndDte As Date) As ADODB.Recordset
    On Error GoTo CreateOrgProdLineDiscErr

    Dim lDiscId As Long
    Dim lRtn As Long
    Dim oBSeq As DawnSeq.BSequence
    Dim oDB As DBOrgProdLineDisc
    Dim sSQL As String
    Dim oRS As ADODB.Recordset

    'Make sure the DiscTypCd and DiscStusCd are valid.
    asDiscTypCd = Trim(asDiscTypCd)
    asDiscStusCd = Trim(asDiscStusCd)
    If asDiscTypCd <> CStr(DiscountTypeCode.ecDollar) And asDiscTypCd <> _
        CStr(DiscountTypeCode.ecPercentage) Then
        RaiseResError DawnProdErrCodes.ecInvalidDiscTypCd, TypeName(Me), _
            "CreateOrgProdLineDisc(...)"
    End If
    If asDiscStusCd <> CStr(DiscountStatusCode.ecActive) And asDiscStusCd <> _
        CStr(DiscountStatusCode.ecInactive) Then
        RaiseResError DawnProdErrCodes.ecInvalidDiscStusCd, TypeName(Me), _
            "CreateOrgProdLineDisc(...)"
    End If

    'make sure that if the discount has an end date, this does not clash with an
    already active discount with an end date
    'this check is for special discounts
    If addDiscEndDte > 0 Then
        sSQL = "SELECT 1 FROM org_prod_line_disc WHERE disc_start_dte is not null
and disc_end_dte is not null "
        sSQL = sSQL & " AND org_id = " & CStr(aOrgID) & " AND prod_line_id = " &
CStr(aProdLineID)
        sSQL = sSQL & " AND (" & GetSQLDate(addDiscStartDte) & " BETWEEN
disc_start_dte AND disc_end_dte "
        sSQL = sSQL & " OR " & GetSQLDate(addDiscEndDte) & " BETWEEN disc_start_dte
AND disc_end_dte "
        sSQL = sSQL & " OR disc_start_dte BETWEEN " & GetSQLDate(addDiscStartDte) &
" AND " & GetSQLDate(addDiscEndDte)
        sSQL = sSQL & " OR disc_end_dte BETWEEN " & GetSQLDate(addDiscStartDte) & "
AND " & GetSQLDate(addDiscEndDte) & " ) "
        sSQL = sSQL & " AND disc_stus_cd = '1' "
        Set oRS = GetSQLRecordset(sSQL)
        If oRS.RecordCount > 0 Then
            RaiseResError DawnProd.DawnProdErrCodes.ecDuplicateSpecial,
                TypeName(Me), "CreateOrgProdLineDisc"
        End If
    Else
        sSQL = " SELECT 1 FROM org_prod_line_disc WHERE disc_start_dte is NOT NULL
AND disc_end_dte IS NULL "
        sSQL = sSQL & " AND org_id = " & CStr(aOrgID) & " AND prod_line_id = " &
CStr(aProdLineID)
        sSQL = sSQL & " AND disc_start_dte = " & GetSQLDate(addDiscStartDte)
    End If
End Function

```

```

                                BProd.cls
    Set oRS = GetSQLRecordset(ssSQL)
    If oRS.RecordCount > 0 Then
        RaiseResError DawnProd.DawnProdErrCodes.ecDuplicateDisc, TypeName(Me),
"CreateOrgProdLineDisc"
    End If
End If

'Get the next disc_id.
Set oBSeq = CtxCreateObject("DawnSeq.BSequence")
lDiscId = oBSeq.GetNextInSequence("discount")
If lDiscId < 1 Then
    RaiseResError DawnProdErrCodes.ecSequenceError, TypeName(Me),
"CreateOrgProdLineDisc(...)"
End If

'Allow the DB component to do the actual insert.
Set oDB = CtxCreateObject("DawnProd.DBOrgProdLineDisc")
lRtn = oDB.CreateOrgProdLineDisc(alOrgID, alProdLineID, lDiscId, asDiscStusCd, _
    asDiscTypCd, adbDiscVal, addDiscStartDte, addDiscEndDte)

If lRtn <> 1 Then 'Something went wrong with the insert
    RaiseResError DawnProdErrCodes.ecInsertFailed, TypeName(Me),
"CreateOrgProdLineDisc(...)"
Else
    'Retrieve the newly created record.
    ssSQL = "SELECT org_id, prod_line_id, disc_id, disc_stus_cd, disc_typ_cd,
disc_val," & _
"disc_start_dte, disc_end_dte, last_maint_dtm FROM org_prod_line_disc"
& _
" WHERE org_id = " & CStr(alOrgID) & " AND prod_line_id = " &
CStr(alProdLineID) & _
" AND disc_id = " & CStr(lDiscId)

    Set CreateOrgProdLineDisc = GetSQLRecordset(ssSQL)
End If

'Tell MTS we're done.
CtxSetComplete

'Release object references
Set oDB = Nothing
Set oBSeq = Nothing
Set oRS = Nothing

Exit Function

CreateOrgProdLineDiscErr:
'Release object references
Set oDB = Nothing
Set oBSeq = Nothing
Set oRS = Nothing

'Set return value

'Abort the transaction
CtxSetAbort

'Raise the error to the calling module
RaiseError TypeName(Me), "CreateOrgProdLineDisc(...)"

Exit Function

End Function

```

BProd.cls

```

'=====
'Updates a discount for a product line for an organization.
'2000/01/28 chrubecky added extra transaction complete before stored procedure
call.
'=====
Public Function UpdOrgProdLineDisc(ByVal alOrgID As Long, ByVal alProdLineID As
Long, ByVal alDiscID As Long, _
ByVal asDiscStusCd As String, ByVal asDiscTypCd As String, ByVal adbDiscVal
As Double, ByVal addDiscStartDte As Date, _
ByVal addDiscEndDte As Date, ByVal adtLastMaintDtm As Date) As
ADODB.Recordset
On Error GoTo UpdOrgProdLineDiscErr

Dim ODB As DBOrgProdLineDisc
Dim lRtn As Long
Dim sSQL As String
Dim oRS As ADODB.Recordset
Dim oRSDisc As ADODB.Recordset

'Make sure the DiscTypCd and DiscStusCd are valid.
asDiscTypCd = Trim(asDiscTypCd)
asDiscStusCd = Trim(asDiscStusCd)
If asDiscTypCd <> CStr(DiscountTypeCode.ecDollar) And asDiscTypCd <> _
CStr(DiscountTypeCode.ecPercentage) Then
RaiseResError DawnProdErrCodes.ecInvalidDiscTypCd, TypeName(Me), _
"UpdOrgProdLineDisc(...)"
End If
If asDiscStusCd <> CStr(DiscountStatusCode.ecActive) And asDiscStusCd <> _
CStr(DiscountStatusCode.ecInactive) Then
RaiseResError DawnProdErrCodes.ecInvalidDiscStusCd, TypeName(Me), _
"UpdOrgProdLineDisc(...)"
End If

'make sure that if the discount has an end date, this does not clash with an
already active discount with an end date
'this check is for special discounts
If asDiscStusCd = ecActive Then
If addDiscEndDte > 0 Then
sSQL = "SELECT 1 FROM org_prod_line_disc WHERE disc_start_dte is not
null and disc_end_dte is not null "
sSQL = sSQL & " AND org_id = " & CStr(alOrgID) & " AND prod_line_id = "
& CStr(alProdLineID)
sSQL = sSQL & " AND disc_id <> " & CStr(alDiscID)
sSQL = sSQL & " AND (" & GetSQLDate(addDiscStartDte) & " BETWEEN
disc_start_dte AND disc_end_dte "
sSQL = sSQL & " OR " & GetSQLDate(addDiscEndDte) & " BETWEEN
disc_start_dte AND disc_end_dte "
sSQL = sSQL & " OR disc_start_dte BETWEEN " &
GetSQLDate(addDiscStartDte) & " AND " & GetSQLDate(addDiscEndDte)
& " AND " & GetSQLDate(addDiscEndDte) & " ) "
sSQL = sSQL & " AND disc_stus_cd = '1' "
Set oRSDisc = GetSQLRecordset(sSQL)
If oRSDisc.RecordCount > 0 Then
RaiseResError DawnProdErrCodes.ecDuplicateSpecial,
TypeName(Me), "UpdOrgProdLineDisc"
End If
Else
sSQL = " SELECT 1 FROM org_prod_line_disc WHERE disc_start_dte is NOT
NULL AND disc_end_dte IS NULL "

```

```

        BProd.cls
        sSQL = sSQL & " AND org_id = " & CStr(alOrgID) & " AND prod_line_id = "
& CStr(alProdLineID)
        sSQL = sSQL & " AND disc_start_dte = " & GetSQLDate(addDiscStartDte)
        sSQL = sSQL & " AND disc_id <> " & CStr(alDiscId)
        Set oRS = GetSQLRecordset(sSQL)
        If oRS.RecordCount > 0 Then
            RaiseResError DawnProd.DawnProdErrCodes.ecDuplicatedDisc,
TypeName(Me), "UpdOrgProdLineDisc"
        End If
    End If
End If

'Allow the DB component to do the actual update.
Set oDB = CtxCreateObject("DawnProd.DBOrgProdLineDisc")
lRtn = oDB.UpdOrgProdLineDisc(alOrgID, alProdLineID, alDiscId, asDiscStusCd,
asDiscTypCd, _
adbdiscVal, addDiscStartDte, addDiscEndDte, addLastMaintDtm)
If lRtn <> 1 Then 'Something went wrong on the update.
    sSQL = "SELECT Count(*) as rtn FROM org_prod_line_disc" & _
    " WHERE org_id = " & CStr(alOrgID) & " AND prod_line_id = " &
CStr(alProdLineID) & _
    " AND disc_id = " & CStr(alDiscId)
    Set oRS = GetSQLRecordset(sSQL)
    If Not oRS.EOF Then
        If oRS.Fields("rtn").Value = 0 Then
            RaiseResError DawnProdErrCodes.ecRecordDeleted, TypeName(Me),
"UpdOrgProdLineDisc(...)"
        Else 'Row changed between retrieve and update.
            RaiseResError DawnProdErrCodes.ecRecordChanged, TypeName(Me),
"UpdOrgProdLineDisc(...)"
        End If
    End If
Else
    'Now get the newly updated record.
    sSQL = "SELECT org_id, prod_line_id, disc_id, disc_stus_cd, disc_typ_cd,
disc_val, " & _
    " disc_start_dte, disc_end_dte, last_maint_dtm FROM org_prod_line_disc"
& _
    " WHERE org_id = " & CStr(alOrgID) & " AND prod_line_id = " &
CStr(alProdLineID) & _
    " AND disc_id = " & CStr(alDiscId)
    Set UpdOrgProdLineDisc = GetSQLRecordset(sSQL)
End If

'Tell MTS the transaction is complete.
CtxSetComplete

'Release object references
Set oDB = Nothing
Set oRS = Nothing
Set oRSDisc = Nothing

Exit Function

UpdOrgProdLineDiscErr:
'Release object references
Set oDB = Nothing
Set oRS = Nothing
Set oRSDisc = Nothing

'Set return value
'Abort the transaction

```

CtxSetAbort

'Raise the error to the calling module
RaiseError TypeName(Me), "UpdOrgProdLineDisc(...)"

Exit Function

End Function

'=====

'Deletes a discount for a product line for an organization.
'2000/01/28 chrubecky added extra transaction complete before stored procedure
call.
'=====

Public Function DelOrgProdLineDisc(ByVal aOrgID As Long, ByVal aProdLineID As
Long, ByVal aDiscID As Long, _
ByVal adtLastMaintDtm As Date)
On Error GoTo DelOrgProdLineDiscErr

Dim ODB As DBOrgProdLineDisc
Dim lRtn As Long
Dim sSQL As String
Dim oRS As ADODB.Recordset

'Allow the DB component to do the actual deletion.
Set ODB = CtxCreateObject("DawnProd.DBOrgProdLineDisc")
lRtn = ODB.DelOrgProdLineDisc(aOrgID, aProdLineID, aDiscID, adtLastMaintDtm)
If lRtn <> 1 Then 'something went wrong
sSQL = "SELECT Count(*) as rtn FROM org_prod_line_disc" & _
" WHERE org_id = " & CStr(aOrgID) & " AND prod_line_id = " &
CStr(aProdLineID) & _
" AND disc_id = " & CStr(aDiscID)
Set oRS = GetSQLRecordset(sSQL)
If Not oRS.EOF Then
If oRS.Fields("rtn").Value = 0 Then
RaiseResError DawnProdErrCodes.ecRecordAlreadyDeleted, TypeName(Me),
"DelOrgProdLineDisc(...)"
Else 'Row changed between retrieve and delete.
RaiseResError DawnProdErrCodes.ecRecordChanged, TypeName(Me),
"DelOrgProdLineDisc(...)"
End If
End If
End If

'Tell MTS the transaction is complete.
CtxSetComplete

'Release object references
Set ODB = Nothing
Set oRS = Nothing

Exit Function

DelOrgProdLineDiscErr:
'Release object references
Set ODB = Nothing
Set oRS = Nothing

'Set return value

'Abort the transaction

CtxSetAbort

'Raise the error to the calling module
RaiseError TypeName(Me), "DelOrgProdLineDisc(...)"

Exit Function

End Function

```
'=====
'Adds a discount for a commodity for an organization.
'2000/01/28 chrubecky added extra transaction complete before stored procedure
call.
'=====
```

```
Public Function CreateOrgCommDisc(ByVal alOrgID As Long, ByVal alCommId As Long, _
    ByVal asDiscStusCd As String, ByVal asDiscTypCd As String, ByVal adbbdiscval
As Double, ByVal addDiscStartDte As Date, _
    ByVal addDiscEndDte As Date) As ADODB.Recordset
    On Error GoTo CreateOrgCommDiscErr
```

```
Dim lDiscId As Long
Dim lRtn As Long
Dim oBSeq As DawnSeq.BSequence
Dim ODB As DBOrgCommDisc
Dim sSQL As String
Dim oRS As ADODB.Recordset
```

```
'Make sure the DiscTypCd and DiscStusCd are valid.
asDiscTypCd = Trim(asDiscTypCd)
asDiscStusCd = Trim(asDiscStusCd)
If asDiscTypCd <> CStr(DiscountTypeCode.ecDollar) And asDiscTypCd <> _
    CStr(DiscountTypeCode.ecPercentage) Then
    RaiseResError DawnProdErrCodes.ecInvalidDiscTypCd, TypeName(Me), _
        "CreateOrgCommDisc(...)"
```

```
End If
If asDiscStusCd <> CStr(DiscountStatusCode.ecActive) And asDiscStusCd <> _
    CStr(DiscountStatusCode.ecInactive) Then
    RaiseResError DawnProdErrCodes.ecInvalidDiscStusCd, TypeName(Me), _
        "CreateOrgCommDisc(...)"
```

End If

'make sure that if the discount has an end date, this does not clash with an
already active discount with an end date

'this check is for special discounts

```
If addDiscEndDte > 0 Then
    sSQL = "SELECT 1 FROM org_comm_disc WHERE disc_start_dte is not null and
disc_end_dte is not null "
    sSQL = sSQL & " AND org_id = " & CStr(alOrgID) & " AND commodity_id = " &
CStr(alCommId)
    sSQL = sSQL & " AND (" & GetSQLDate(adDiscStartDte) & " BETWEEN
disc_start_dte AND disc_end_dte "
    sSQL = sSQL & " OR " & GetSQLDate(adDiscEndDte) & " BETWEEN disc_start_dte
AND disc_end_dte "
    sSQL = sSQL & " OR disc_start_dte BETWEEN " & GetSQLDate(adDiscStartDte) &
" AND " & GetSQLDate(adDiscEndDte)
    sSQL = sSQL & " OR disc_end_dte BETWEEN " & GetSQLDate(adDiscStartDte) &
AND " & GetSQLDate(adDiscEndDte) & " ) "
    sSQL = sSQL & " AND disc_stus_cd = '1' "
    Set oRS = GetSQLRecordset(sSQL)
    If oRS.RecordCount > 0 Then
        RaiseResError DawnProd.DawnProdErrCodes.ecDuplicateSpecial,
```

```

TypeName(Me), "CreateOrgCommDisc"
End If
Else
    sSQL = " SELECT 1 FROM org_comm_disc WHERE disc_start_dte IS NOT NULL AND
disc_end_dte IS NULL "
    sSQL = sSQL & " AND org_id = " & CStr(alOrgID) & " AND commodity_id = " &
CStr(alCommId)
    sSQL = sSQL & " AND disc_start_dte = " & GetSQLDate(adDiscStartDte)
    Set oRS = GetSQLRecordset(sSQL)
    If oRS.RecordCount > 0 Then
        RaiseResError DawnProd.DawnProdErrCodes.ecDuplicateDisc, TypeName(Me),
"CreateOrgCommDisc"
    End If
End If

'Get the next disc_id.
Set oBSeq = CtxCreateObject("DawnSeq.BSequence")
lDiscId = oBSeq.GetNextInSequence("discount")
If lDiscId < 1 Then
    RaiseResError DawnProdErrCodes.ecSequenceError, TypeName(Me),
"CreateOrgCommDisc(...)"
End If

'Allow the DB component to do the actual insert.
Set oDB = CtxCreateObject("DawnProd.DBOrgCommDisc")
lRtn = oDB.CreateOrgCommDisc(alOrgID, alCommId, lDiscId, asDiscStusCd, _
asDiscTypCd, adbDiscVal, adDiscStartDte, adDiscEndDte)

If lRtn <> 1 Then 'Something went wrong with the insert
    RaiseResError DawnProdErrCodes.ecInsertFailed, TypeName(Me),
"CreateOrgCommDisc(...)"
Else
    'Retrieve the newly created record.
    sSQL = "SELECT org_id, commodity_id, disc_id, disc_stus_cd, disc_typ_cd,
disc_val," & _
" disc_start_dte, disc_end_dte, last_maint_dtm FROM org_comm_disc" & _
" WHERE org_id = " & CStr(alOrgID) & " AND commodity_id = " &
CStr(alCommId) & _
" AND disc_id = " & CStr(lDiscId)

    Set CreateOrgCommDisc = GetSQLRecordset(sSQL)
End If

'Tell MTS we're done.
CtxSetComplete

'Release object references
Set oDB = Nothing
Set oBSeq = Nothing
Set oRS = Nothing

Exit Function

CreateOrgCommDiscErr:
'Release object references
Set oDB = Nothing
Set oBSeq = Nothing
Set oRS = Nothing

'Set return value

'Abort the transaction
CtxSetAbort

```

BProd.cls

```
'Raise the error to the calling module
RaiseError TypeName(Me), "CreateOrgCommDisc(...)"
```

```
Exit Function
```

```
End Function
```

```
'=====
'Updates a discount for a commodity for an organization.
'2000/01/28 chrubecky added extra transaction complete before stored procedure
call.
'=====
Public Function UpdOrgCommDisc(ByVal aOrgID As Long, ByVal aCommId As Long, ByVal
aDiscId As Long, _
ByVal asDiscStusCd As String, ByVal asDiscTypCd As String, ByVal adbDiscVal
As Double, ByVal addDiscStartDte As Date, _
ByVal addDiscEndDte As Date, ByVal adtLastMaintDtm As Date) As
ADODB.Recordset
On Error GoTo UpdOrgCommDiscErr

Dim oDB As DBOrgCommDisc
Dim lRtn As Long
Dim sSQL As String
Dim oRS As ADODB.Recordset
Dim oRSDisc As ADODB.Recordset

'Make sure the DiscTypCd and DiscStusCd are valid.
asDiscTypCd = Trim(asDiscTypCd)
asDiscStusCd = Trim(asDiscStusCd)
If asDiscTypCd <> CStr(DiscountTypeCode.ecDollar) And asDiscTypCd <> _
CStr(DiscountTypeCode.ecPercentage) Then
RaiseResError DawnProdErrCodes.ecInvalidDiscTypCd, TypeName(Me), _
"UpdOrgCommDisc(...)"
End If
If asDiscStusCd <> CStr(DiscountStatusCode.ecActive) And asDiscStusCd <> _
CStr(DiscountStatusCode.ecInactive) Then
RaiseResError DawnProdErrCodes.ecInvalidDiscStusCd, TypeName(Me), _
"UpdOrgCommDisc(...)"
End If

'make sure that if the discount has an end date, this does not clash with an
already active discount with an end date
'this check is for special discounts
If asDiscStusCd = ecActive Then
If addDiscEndDte > 0 Then
sSQL = "SELECT 1 FROM org_comm_disc WHERE disc_start_dte is not null and
disc_end_dte is not null "
sSQL = sSQL & " AND org_id = " & CStr(aOrgID) & " AND commodity_id = "
& CStr(aCommId)
sSQL = sSQL & " AND disc_id <> " & CStr(aDiscId)
sSQL = sSQL & " AND (" & GetSQLDate(addDiscStartDte) & " BETWEEN
disc_start_dte AND disc_end_dte "
sSQL = sSQL & " OR " & GetSQLDate(addDiscEndDte) & " BETWEEN
disc_start_dte AND disc_end_dte "
sSQL = sSQL & " OR disc_start_dte BETWEEN " &
GetSQLDate(addDiscStartDte) & " AND " & GetSQLDate(addDiscEndDte)
sSQL = sSQL & " OR disc_end_dte BETWEEN " & GetSQLDate(addDiscStartDte)
& " AND " & GetSQLDate(addDiscEndDte) & " ) "
sSQL = sSQL & " AND disc_stus_cd = '1' "
Set oRSDisc = GetSQLRecordset(sSQL)
```

```

                                BProd.cls
    If ORSDisc.RecordCount > 0 Then
        RaiseResError DawnProd.DawnProdErrCodes.ecDuplicatesSpecial,
        TypeName(Me), "UpdOrgCommDisc"
    End If
    Else
        sSQL = " SELECT 1 FROM org_comm_disc WHERE disc_start_dte is NOT NULL
AND disc_end_dte IS NULL "
        sSQL = sSQL & " AND org_id = " & CStr(alOrgID) & " AND commodity_id = "
& CStr(alCommId)
        sSQL = sSQL & " AND disc_start_dte = " & GetSQLDate(adDiscStartDte)
        sSQL = sSQL & " AND disc_id <> " & CStr(alDiscId)
        Set oRS = GetSQLRecordset(sSQL)
        If oRS.RecordCount > 0 Then
            RaiseResError DawnProd.DawnProdErrCodes.ecDuplicatedDisc,
            TypeName(Me), "UpdOrgCommDisc"
        End If
    End If
End If

'Allow the DB component to do the actual insert.
Set oDB = CtxCreateObject("DawnProd.DBOrgCommDisc")
lRtn = oDB.UpdOrgCommDisc(alOrgID, alCommId, alDiscId, asDiscStusCd,
asDiscTypCd, _
adbdiscVal, adDiscStartDte, adDiscEndDte, adtLastMaintDtm)
If lRtn <> 1 Then 'Something went wrong on the update.
    sSQL = "SELECT Count(*) as rtn FROM org_comm_disc" & _
    " WHERE org_id = " & CStr(alOrgID) & " AND commodity_id = " &
CStr(alCommId) & _
    " AND disc_id = " & CStr(alDiscId)
    Set oRS = GetSQLRecordset(sSQL)
    If Not oRS.EOF Then
        If oRS.Fields("rtn").Value = 0 Then
            RaiseResError DawnProdErrCodes.ecRecordDeleted, TypeName(Me);
            "UpdOrgCommDisc(...)"
        Else 'Row changed between retrieve and update.
            RaiseResError DawnProdErrCodes.ecRecordChanged, TypeName(Me),
            "UpdOrgCommDisc(...)"
        End If
    End If
Else
    'Now get the newly updated record.
    sSQL = "SELECT org_id, commodity_id, disc_id, disc_stus_cd, disc_typ_cd,
disc_val," & _
    " disc_start_dte, disc_end_dte, last_maint_dtm FROM org_comm_disc" & _
    " WHERE org_id = " & CStr(alOrgID) & " AND commodity_id = " &
CStr(alCommId) & _
    " AND disc_id = " & CStr(alDiscId)
    Set UpdOrgCommDisc = GetSQLRecordset(sSQL)
End If

'Tell MTS the transaction is complete.
CtxSetComplete

'Release object references
Set oDB = Nothing
Set oRS = Nothing
Set ORSDisc = Nothing

Exit Function

UpdOrgCommDiscErr:
'Release object references
Set oDB = Nothing

```

BProd.cls

```

Set oRS = Nothing
Set oRSDisc = Nothing

'Set return value

'Abort the transaction
CtxSetAbort

'Raise the error to the calling module
RaiseError TypeName(Me), "UpdOrgCommDisc(...)"

Exit Function

End Function

'=====
'Deletes a discount for a commodity for an organization.
'2000/01/28 chrubecky added extra transaction complete before stored procedure
call.
'=====
Public Function DelOrgCommDisc(ByVal alOrgID As Long, ByVal alCommId As Long, ByVal
alDiscId As Long, _
ByVal adtLastMaintDtm As Date)
On Error GoTo DelOrgCommDiscErr

Dim oDB As DBOrgCommDisc
Dim lRtn As Long
Dim sSQL As String
Dim oRS As ADODB.Recordset

'Allow the DB component to do the actual deletion.
Set oDB = CtxCreateObject("DawnProd.DBOrgCommDisc")
lRtn = oDB.DelOrgCommDisc(alOrgID, alCommId, alDiscId, adtLastMaintDtm)
If lRtn <> 1 Then 'something went wrong
    sSQL = "SELECT Count(*) as rtn FROM org_comm_disc" & _
        " WHERE org_id = " & CStr(alOrgID) & " AND commodity_id = " &
CStr(alCommId) & _
        " AND disc_id = " & CStr(alDiscId)
    Set oRS = GetSQLRecordset(sSQL)
    If Not oRS.EOF Then
        If oRS.Fields("rtn").Value = 0 Then
            RaiseResError DawnProdErrCodes.ecRecordAlreadyDeleted, TypeName(Me),
"DelOrgCommDisc(...)"
        Else 'Row changed between retrieve and delete.
            RaiseResError DawnProdErrCodes.ecRecordChanged, TypeName(Me),
"DelOrgCommDisc(...)"
        End If
    End If
End If

'Tell MTS the transaction is complete.
CtxSetComplete

'Release object references
Set oDB = Nothing
Set oRS = Nothing

Exit Function

DelOrgCommDiscErr:
'Release object references

```

BProd.cls

```

Set oDB = Nothing
Set oRS = Nothing

'Set return value
'Abort the transaction
CtxSetAbort

'Raise the error to the calling module
RaiseError TypeName(Me), "DelOrgCommDisc(...)"

Exit Function

End Function

'=====
'Adds a discount for a part for an organization.
'=====
Public Function CreateOrgPartDisc(ByVal alOrgID As Long, ByVal alPartID As Long, _
    ByVal asDiscStusCd As String, ByVal asDiscTypCd As String, ByVal adbDiscVal _
    As Double, ByVal addDiscStartDte As Date, _
    ByVal addDiscEndDte As Date) As ADODB.Recordset
    On Error GoTo CreateOrgPartDiscErr

    Dim lDiscID As Long
    Dim lRtn As Long
    Dim oBSeq As DawnSeq.BSequence
    Dim oDB As DBOrgPartDisc
    Dim sSQL As String
    Dim oRS As ADODB.Recordset

    'Make sure the DiscTypCd and DiscStusCd are valid.
    asDiscTypCd = Trim(asDiscTypCd)
    asDiscStusCd = Trim(asDiscStusCd)
    If asDiscTypCd <> CStr(DiscountTypeCode.ecDollar) And asDiscTypCd <> _
        CStr(DiscountTypeCode.ecPercentage) Then
        RaiseResError DawnProdErrCodes.ecInvalidDiscTypCd, TypeName(Me), _
            "CreateOrgPartDisc(...)"
    End If
    If asDiscStusCd <> CStr(DiscountStatusCode.ecActive) And asDiscStusCd <> _
        CStr(DiscountStatusCode.ecInactive) Then
        RaiseResError DawnProdErrCodes.ecInvalidDiscStusCd, TypeName(Me), _
            "CreateOrgPartDisc(...)"
    End If

    'make sure that if the discount has an end date, this does not clash with an
    already active discount with an end date
    'this check is for special discounts
    If addDiscEndDte > 0 Then
        sSQL = "SELECT 1 FROM org_part_disc WHERE disc_start_dte is not null and
disc_end_dte is not null "
        sSQL = sSQL & " AND org_id = " & CStr(alOrgID) & " AND part_id = " &
CStr(alPartID)
        sSQL = sSQL & " AND (" & GetSQLDate(addDiscStartDte) & " BETWEEN
disc_start_dte AND disc_end_dte "
        sSQL = sSQL & " OR " & GetSQLDate(addDiscEndDte) & " BETWEEN disc_start_dte
AND disc_end_dte "
        sSQL = sSQL & " OR disc_start_dte BETWEEN " & GetSQLDate(addDiscStartDte) &
" AND " & GetSQLDate(addDiscEndDte)
        sSQL = sSQL & " OR disc_end_dte BETWEEN " & GetSQLDate(addDiscStartDte) & "
AND " & GetSQLDate(addDiscEndDte) & " ) "
    End If

```

```

                                BProd.cls
    sSQL = sSQL & " AND disc_stus_cd = '1' "
    Set ORS = GetSQLRecordset(sSQL)
    If ORS.RecordCount > 0 Then
        RaiseResError DawnProd.DawnProdErrCodes.ecDuplicateSpecial,
TypeName(Me), "CreateOrgPartDisc"
    End If
Else
    sSQL = " SELECT 1 FROM org_part_disc WHERE disc_start_dte is NOT NULL AND
disc_end_dte IS NULL "
    sSQL = sSQL & " AND org_id = " & CStr(alOrgID) & " AND part_id = " &
CStr(alPartID)
    sSQL = sSQL & " AND disc_start_dte = " & GetSQLDate(adDiscStartDte)
    Set ORS = GetSQLRecordset(sSQL)
    If ORS.RecordCount > 0 Then
        RaiseResError DawnProd.DawnProdErrCodes.ecDuplicateDisc, TypeName(Me),
"CreateOrgPartDisc"
    End If
End If

'Get the next disc_id.
Set oBSeq = CtxCreateObject("DawnSeq.BSequence")
lDiscId = oBSeq.GetNextInSequence("discount")
If lDiscId < 1 Then
    RaiseResError DawnProdErrCodes.ecSequenceError, TypeName(Me),
"CreateOrgPartDisc(...)"
End If

'Allow the DB component to do the actual insert.
Set oDB = CtxCreateObject("DawnProd.DBOrgPartDisc")
lRtn = oDB.CreateOrgPartDisc(alOrgID, alPartID, lDiscId, asDiscStusCd, _
asDiscTypCd, adbDiscVal, adDiscStartDte, adDiscEndDte)

If lRtn <> 1 Then 'Something went wrong with the insert
    RaiseResError DawnProdErrCodes.ecInsertFailed, TypeName(Me),
"CreateOrgPartDisc(...)"
Else
    'Retrieve the newly created record.
    sSQL = "SELECT org_id, part_id, disc_id, disc_stus_cd, disc_typ_cd,
disc_val," & _
        " disc_start_dte, disc_end_dte, last_maint_dtm FROM org_part_disc" & _
        " WHERE org_id = " & CStr(alOrgID) & " AND part_id = " & CStr(alPartID)
& _
        " AND disc_id = " & CStr(lDiscId)

    Set CreateOrgPartDisc = GetSQLRecordset(sSQL)
End If

'Tell MTS we're done.
CtxSetComplete

'Release object references
Set oDB = Nothing
Set oBSeq = Nothing
Set ORS = Nothing

Exit Function

CreateOrgPartDiscErr:
'Release object references
Set oDB = Nothing
Set oBSeq = Nothing
Set ORS = Nothing

```

```

'Set return value

'Abort the transaction
CtxSetAbort

'Raise the error to the calling module
RaiseError TypeName(Me), "CreateOrgPartDisc(...)"

Exit Function

End Function

'=====
'Updates a discount for a part for an organization.
'=====
Public Function UpdOrgPartDisc(ByVal alOrgID As Long, ByVal alPartID As Long, ByVal alDiscID As Long, _
    ByVal asDiscStusCd As String, ByVal asDiscTypCd As String, ByVal adbDiscVal As Double, ByVal addiscStartDte As Date, _
    ByVal addiscEndDte As Date, ByVal adtLastMaintDtm As Date) As ADODB.Recordset
    On Error GoTo UpdOrgPartDiscErr

    Dim oDB As DBOrgPartDisc
    Dim lRtn As Long
    Dim sSQL As String
    Dim oRS As ADODB.Recordset
    Dim oRSDisc As ADODB.Recordset

    'Make sure the DiscTypCd and DiscStusCd are valid.
    asDiscTypCd = Trim(asDiscTypCd)
    asDiscStusCd = Trim(asDiscStusCd)
    If asDiscTypCd <> CStr(DiscountTypeCode.ecDollar) And asDiscTypCd <> _
        CStr(DiscountTypeCode.ecPercentage) Then
        RaiseResError DawnProdErrCodes.ecInvalidDiscTypCd, TypeName(Me), _
            "UpdOrgPartDisc(...)"
    End If
    If asDiscStusCd <> CStr(DiscountStatusCode.ecActive) And asDiscStusCd <> _
        CStr(DiscountStatusCode.ecInactive) Then
        RaiseResError DawnProdErrCodes.ecInvalidDiscStusCd, TypeName(Me), _
            "UpdOrgPartDisc(...)"
    End If

    'make sure that if the discount has an end date, this does not clash with an
    already active discount with an end date
    'this check is for special discounts
    If asDiscStusCd = ecActive Then
        If addiscEndDte > 0 Then
            sSQL = "SELECT 1 FROM org_part_disc WHERE disc_start_dte is not null and
disc_end_dte is not null "
            sSQL = sSQL & " AND org_id = " & CStr(alOrgID) & " AND part_id = " &
CStr(alPartID)
            sSQL = sSQL & " AND disc_id <> " & CStr(alDiscID)
            sSQL = sSQL & " AND (" & GetSQLDate(addiscStartDte) & " BETWEEN
disc_start_dte AND disc_end_dte "
            sSQL = sSQL & " OR " & GetSQLDate(addiscEndDte) & " BETWEEN
disc_start_dte AND disc_end_dte "
            sSQL = sSQL & " OR disc_start_dte BETWEEN " &
GetSQLDate(addiscStartDte) & " AND " & GetSQLDate(addiscEndDte)
            sSQL = sSQL & " OR disc_end_dte BETWEEN " & GetSQLDate(addiscStartDte)
& " AND " & GetSQLDate(addiscEndDte) & ")"

```



```

                                BProd.cls
    sSQL = sSQL & " AND disc_stus_cd = '1' "
    Set oRSDisc = GetSQLRecordset(sSQL)
    If oRSDisc.RecordCount > 0 Then
        RaiseResError DawnProd.DawnProdErrCodes.ecDuplicateSpecial,
TypeName(Me), "UpdOrgPartDisc"
    End If
    Else
        sSQL = " SELECT 1 FROM org_part_disc WHERE disc_start_dte is NOT NULL
AND disc_end_dte IS NULL "
        sSQL = sSQL & " AND org_id = " & CStr(alOrgID) & " AND part_id = " &
CStr(alPartID)
        sSQL = sSQL & " AND disc_start_dte = " & GetSQLDate(addDiscStartDte)
        sSQL = sSQL & " AND disc_id <> " & CStr(alDiscId)
        Set oRS = GetSQLRecordset(sSQL)
        If oRS.RecordCount > 0 Then
            RaiseResError DawnProd.DawnProdErrCodes.ecDuplicateDisc,
TypeName(Me), "UpdOrgPartDisc"
        End If
    End If
End If

'Allow the DB component to do the actual update.
Set oDB = CtxCreateObject("DawnProd.DBOrgPartDisc")
lRtn = oDB.UpdOrgPartDisc(alOrgID, alPartID, alDiscId, asDiscStusCd,
asDiscTypCd, _
adbDiscVal, addDiscStartDte, addDiscEndDte, adtLastMaintDtm)
If lRtn <> 1 Then 'Something went wrong on the update.
    sSQL = "SELECT Count(*) as rtn FROM org_part_disc" & _
    " WHERE org_id = " & CStr(alOrgID) & " AND part_id = " & CStr(alPartID)
    & _
    " AND disc_id = " & CStr(alDiscId)
    Set oRS = GetSQLRecordset(sSQL)
    If Not oRS.EOF Then
        If oRS.Fields("rtn").Value = 0 Then
            RaiseResError DawnProdErrCodes.ecRecordDeleted, TypeName(Me),
"UpdOrgPartDisc(...)"
        Else 'Row changed between retrieve and update.
            RaiseResError DawnProdErrCodes.ecRecordChanged, TypeName(Me),
"UpdOrgPartDisc(...)"
        End If
    End If
Else
    'Now get the newly updated record.
    sSQL = "SELECT org_id, part_id, disc_id, disc_stus_cd, disc_typ_cd,
disc_val," & _
    " disc_start_dte, disc_end_dte, last_maint_dtm FROM org_part_disc" & _
    " WHERE org_id = " & CStr(alOrgID) & " AND part_id = " & CStr(alPartID)
    & _
    " AND disc_id = " & CStr(alDiscId)
    Set UpdOrgPartDisc = GetSQLRecordset(sSQL)
End If

'Tell MTS the transaction is complete.
CtxSetComplete

'Release object references
Set oDB = Nothing
Set oRS = Nothing
Set oRSDisc = Nothing

Exit Function
UpdOrgPartDiscErr:

```

BProd.cls

```

'Release object references
Set oDB = Nothing
Set oRS = Nothing
Set oRSDisc = Nothing

'Set return value

'Abort the transaction
CtxSetAbort

'Raise the error to the calling module
RaiseError TypeName(Me), "UpdOrgPartDisc(...)"

Exit Function

End Function

'=====
'Deletes a discount for a part for an organization.
'=====
Public Function DelOrgPartDisc(ByVal aOrgID As Long, ByVal aPartID As Long, ByVal
aDiscID As Long, _
    ByVal adtLastMaintDtm As Date)
    On Error GoTo DelOrgPartDiscErr

    Dim oDB As DBOrgPartDisc
    Dim lRtn As Long
    Dim sSQL As String
    Dim oRS As ADODB.Recordset

    'Allow the DB component to do the actual deletion.
    Set oDB = CtxCreateObject("DawnProd.DBOrgPartDisc")
    lRtn = oDB.DelOrgPartDisc(aOrgID, aPartID, aDiscID, adtLastMaintDtm)
    If lRtn <> 1 Then 'something went wrong
        sSQL = "SELECT Count(*) as rtn FROM org_part_disc" & _
            " WHERE org_id = " & CStr(aOrgID) & " AND part_id = " & CStr(aPartID)
        & _
            " AND disc_id = " & CStr(aDiscID)
        Set oRS = GetSQLRecordset(sSQL)
        If Not oRS.EOF Then
            If oRS.Fields("rtn").Value = 0 Then
                RaiseResError DawnProdErrCodes.ecRecordAlreadyDeleted, TypeName(Me),
                "DelOrgPartDisc(...)"
            Else 'Row changed between retrieve and delete.
                RaiseResError DawnProdErrCodes.ecRecordChanged, TypeName(Me),
                "DelOrgPartDisc(...)"
            End If
        End If
    End If

    'Tell MTS the transaction is complete.
    CtxSetComplete

    'Release object references
    Set oDB = Nothing
    Set oRS = Nothing

    Exit Function

DelOrgPartDiscErr:
'Release object references

```

BProd.cls

```

Set oDB = Nothing
Set oRS = Nothing

'Set return value

'Abort the transaction
CtxSetAbort

'Raise the error to the calling module
RaiseError TypeName(Me), "DelOrgPartDisc(...)"

Exit Function

End Function
'=====
'this functions adds a mfr accepted part to mfr accepted part table
'arguments - mfr ID, source mfr ID = M&R = 0, part ID, commodity ID, prodline ID,
default source ind = 0, available indicator = 1
'returns none
'=====
Public Sub AddManfAcceptPart(ByVal alManfID As Long, ByVal alPartID As Long,
ByVal alCommodityID As Long, ByVal
alProdLineID As Long)
On Error GoTo ErrHandler

'vars
Dim oDBAccept As DawnProd.DBMfrAcceptPart
Dim lRow As Long

'accept ind = 0 - mfr will set this thru update
'default source - from supplier so source id = 0, but default ind = 0 , mfr will
set this thru update
'all supplier parts are always available so avail ind = '1'
'remove part and add part
Set oDBAccept = CtxCreateObject("DawnProd.DBMfrAcceptPart")

lRow = oDBAccept.RemoveMfrAcceptPart(alManfID, alPartID, 0)
If lRow < 0 Then
RaiseResError DawnProd.DawnProdErrCodes.ecDeleteFailed, TypeName(Me),
"AddManfAcceptPart"
End If
'insert row into accept part table
lRow = oDBAccept.AddMfrAcceptPart(alManfID, alPartID, 0, alCommodityID,
alProdLineID, "0", "0", "1")
If lRow <= 0 Then
RaiseResError DawnProd.DawnProdErrCodes.ecInsertFailed, TypeName(Me),
"AddManfAcceptPart"
End If

'run SP for ref table and update the tri state codes
Call RunSP("sp_mfr_accept_part_insert", MakeIntSPParm("aLong", alManfID), _
MakeIntSPParm("aLong",
alProdLineID), _
MakeIntSPParm("aLong",
alCommodityID), _
MakeIntSPParm("aLong",
alPartID))

'run SP for pipelines
Call RunSP("sp_pipeline_mfr_product_add", MakeIntSPParm("aLong", alManfID), _
MakeIntSPParm("aLong", -1), _

```

```

MakeIntSPParm("aLong", -1), _
MakeIntSPParm("aLong", alPartID))

    CtxSetComplete
    Set oDBAccept = Nothing

    Exit Sub

ErrorHandler:
    CtxSetAbort
    Set oDBAccept = Nothing
    RaiseError TypeName(Me), "AddManfAcceptPart"
    Exit Sub

End Sub
'=====
'this function adds an accepted commodity into the mfr accept table
'arguments mfr ID, source = 0 M&R, commodity ID, default source = 0, avilable
'indicator = 1
'returns none
'=====
Public Sub AddManfAcceptCommodity(ByVal alManfID As Long, ByVal alCommodityID As Long)
    On Error GoTo ErrorHandler

    'vars
    Dim oDBAccept As DawnProd.DBMfrAcceptPart
    Dim lRow As Long

    'accept ind = 0 - mfr will set this thru update
    'default source - from supplier so source id = 0, but default ind = 0 , mfr will
    set this thru update
    'all supplier parts are always avilable so avail ind = '1'
    'delete any earlier parts for this commodity and then add the whole commodity

    Set oDBAccept = CtxCreateObject("DawnProd.DBMfrAcceptPart")
    lRow = oDBAccept.RemoveMfrAcceptCommodity(alManfID, alCommodityID)
    If lRow < 0 Then
        RaiseResError DawnProd.DawnProdErrCodes.ecDeleteFailed, TypeName(Me),
        "AddManfAcceptCommodity"
    End If
    lRow = oDBAccept.AddMfrAcceptCommodity(alManfID, 0, alCommodityID, "0", "0",
    "1")
    If lRow <= 0 Then
        RaiseResError DawnProd.DawnProdErrCodes.ecInsertFailed, TypeName(Me),
        "AddManfAcceptCommodity"
    End If

    'run SP for ref table and update the tri state codes
    Call RunSP("sp_mfr_accept_part_insert", MakeIntSPParm("aLong", alManfID), _
        MakeIntSPParm("aLong", -1), _
        _
        MakeIntSPParm("aLong",
        alCommodityID), _
        MakeIntSPParm("aLong", -1))

    'run SP for pipelines
    Call RunSP("sp_pipeline_mfr_product_add", MakeIntSPParm("aLong", alManfID), _
        MakeIntSPParm("aLong", -1), _
        MakeIntSPParm("aLong", alCommodityID), _

```

```
MakeIntSPParm("aLong", -1))
```

```
    CtxSetComplete
    Set oDBAccept = Nothing
```

```
Exit Sub
```

```
ErrorHandler:
```

```
    CtxSetAbort
    Set oDBAccept = Nothing
    RaiseError TypeName(Me), "AddManfAcceptCommodity"
    Exit Sub
```

```
End Sub
```

```
'=====
'this function adds an accepted prod line into the mfr accept part table
'argument - mfr ID, prod line ID, source = 0, default = 0 and avilable innd = 1
'returns none
'=====
```

```
Public Sub AddManfAcceptProdLine(ByVal alManfID As Long, ByVal alProdLineID As Long)
On Error GoTo ErrorHandler
```

```
    'vars
    Dim oDBAccept As DawnProd.DBMfrAcceptPart
    Dim lRow As Long
```

```
    'accept ind = 0 - mfr will set this thru update
    'default source - from supplier so source id = 0, but default ind = 0 , mfr will
set this thru update
    'all supplier parts are always available so avail ind = '1'
    'remove all parts for this prod line and then add all parts for the prod line
```

```
    Set oDBAccept = CtxCreateObject("DawnProd.DBMfrAcceptPart")
    lRow = oDBAccept.RemoveMfrAcceptProdLine(alManfID, alProdLineID)
    If lRow < 0 Then
        RaiseResError DawnProd.DawnProdErrCodes.ecDeleteFailed, TypeName(Me),
"AddManfAcceptProdLine"
    End If
    lRow = oDBAccept.AddMfrAcceptProdLine(alManfID, 0, alProdLineID, "0", "0", "1")
    If lRow <= 0 Then
        RaiseResError DawnProd.DawnProdErrCodes.ecInsertFailed, TypeName(Me),
"AddManfAcceptProdLine"
    End If
```

```
    'run SP for ref table and update the tri state codes
    Call RunSP("sp_mfr_accept_part_insert", MakeIntSPParm("aLong", alManfID), _
                                                    MakeIntSPParm("aLong",
alProdLineID), _
                                                    MakeIntSPParm("aLong", -1),
-
                                                    MakeIntSPParm("aLong", -1))
```

```
    'run SP for pipelines
    Call RunSP("sp_pipeline_mfr_product_add", MakeIntSPParm("aLong", alManfID), _
MakeIntSPParm("aLong", alProdLineID), _
MakeIntSPParm("aLong", -1), _
MakeIntSPParm("aLong", -1))
```

```
    CtxSetComplete
    Set oDBAccept = Nothing
```

Exit Sub

ErrorHandler:

CtxSetAbort

Set oDBAccept = Nothing

RaiseError TypeName(Me), "AddManfAcceptProdLine"

Exit Sub

End Sub

```
'=====
'this function updates the default source indicator and available inndicator for a
part which has been accepted into the
```

```
'mfr accept part table
```

```
'arguments mfr ID, part ID, source ID, default indicator, available indicator
'=====
```

```
Public Sub UpdateManfAcceptPart(ByVal alManfID As Long, ByVal arrPartID As Variant,
ByVal alSourceManfID As Long, _
```

```
ByVal asAcceptInd As String)
```

On Error GoTo ErrorHandler

Dim sPart As String

Dim lCounter As Long

Dim sSQL As String

For lCounter = 0 To UBound(arrPartID)

If lCounter = UBound(arrPartID) Then

sPart = sPart & CStr(arrPartID(lCounter))

Else

sPart = sPart & CStr(arrPartID(lCounter)) & " , "

End If

Next

sPart = " (" & sPart & ") "

'create temp table for use by stored procedure

sSQL = " CREATE table #parts (part_id int)"

RunSQL (sSQL)

'add all the parts for all the commodities in the array into the temp table

sSQL = " INSERT INTO #parts SELECT part_id FROM mfr_accept_part WHERE mfr_id =

" & CStr(alManfID)

```
& sSQL = sSQL & " AND src_mfr_id = " & CStr(alSourceManfID) & " AND part_id in "
```

```
& sPart
RunSQL (sSQL)
```

'run SP to accept and calc price

```
Call RunSP("sp_mfr_accept_part_update_test", MakeIntSPParm("aLong",
alManfID), _
```

```
MakeIntSPParm("aLong", alSourceManfID), _
```

```
MakeStrSPParm("aString", 255, asAcceptInd), _
```

```
MakeStrSPParm("aString", 2, "P"), _
```

```
MakeStrSPParm("aString", 1000, sPart))
```

'vars

Dim oDBAccept As DawnProd.DBMfrAcceptPart

Dim lRow As Long

Set oDBAccept = CtxCreateObject("DawnProd.DBMfrAcceptPart")

```
lRow = oDBAccept.UpdateMfrAcceptPart(alManfID, alPartID, alSourceManfID,
asAcceptInd, asDefaultSourceInd, asAvailableInd)
```

BProd.cls

```

' If lRow <= 0 Then
'     RaiseResError DawnProd.DawnProdErrCodes.ecInsertFailed, TypeName(Me),
"UpdateManfAcceptPart"
' End If
'
' 'run SP for ref table and update the tri state codes
' Call RunSP("sp_mfr_accept_part_update", MakeIntSPParm("aLong", alManfID), _
MakeIntSPParm("aLong", alSourceManfID), _
MakeIntSPParm("aLong", -1), _
MakeIntSPParm("aLong", -1), _
MakeIntSPParm("aLong", alPartID), _
MakeStrSPParm("aString", 255, asAcceptInd))
' 'run SP for pipelines
' If asAcceptInd = 1 Then
'     Call RunSP("sp_pipeline_mfr_product_add", MakeIntSPParm("aLong", alManfID),
'
' MakeIntSPParm("aLong", -1), _
' MakeIntSPParm("aLong", -1), _
' MakeIntSPParm("aLong", alPartID))
' Else
'     Call RunSP("sp_pipeline_mfr_product_remove", MakeIntSPParm("aLong",
alManfID), _
' MakeIntSPParm("aLong", -1), _
' MakeIntSPParm("aLong", -1), _
' MakeIntSPParm("aLong", alPartID))
' End If
' Set oDBAccept = Nothing
' CtxSetComplete
' Exit Sub

ErrHandler:
' Set oDBAccept = Nothing
' CtxSetAbort
' RaiseError TypeName(Me), "UpdateManfAcceptPart"
' Exit Sub

End Sub
'=====
' this function updates the accept ind for a commodity and its parts which has been
accepted into the
' mfr accept part table
' arguments mfr ID, part ID, source ID, default indicator, available indicator
' returns none
'=====
Public Sub UpdateManfAcceptCommodity(ByVal alManfID As Long, ByVal alSourceManfID As
Long, _
ByVal arrCommodityID As
Variant, ByVal asAcceptInd As String)
On Error GoTo ErrHandler

Dim sComm As String
Dim lCounter As Long

```

```

Dim sSQL As String

For lCounter = 0 To UBound(arrCommodityID)
    If lCounter = UBound(arrCommodityID) Then
        sComm = sComm & CStr(arrCommodityID(lCounter))
    Else
        sComm = sComm & CStr(arrCommodityID(lCounter)) & " , "
    End If
Next
sComm = " ( " & sComm & " ) "

'create temp table for use by stored procedure
sSQL = " CREATE table #parts (part_id int)"
RunSQL (sSQL)

'add all the parts for all the commodities in the array into the temp table
sSQL = " INSERT INTO #parts  SELECT part_id FROM mfr_accept_part WHERE mfr_id = " & CStr(alManfID)
sSQL = sSQL & " AND src_mfr_id = " & CStr(alSourceManfID) & " AND commodity_id in " & sComm
RunSQL (sSQL)

'run SP to accept and calc price
Call RunSP("sp_mfr_accept_part_update_test", MakeIntSPParm("aLong", alManfID), _
MakeIntSPParm("aLong", alSourceManfID), _
MakeStrSPParm("aString", 255, asAcceptInd), _
MakeStrSPParm("aString", 2, "C"), _
MakeStrSPParm("aString", 1000, sComm))

'vars
Dim oDBAccept As DawnProd.DBMfrAcceptPart
Dim lRow As Long

Set oDBAccept = CtxCreateObject("DawnProd.DBMfrAcceptPart")
lRow = oDBAccept.UpdateMfrAcceptCommodity(alManfID, alSourceManfID, alCommodityID, asAcceptInd)
If lRow <= 0 Then
    RaiseResError DawnProd.DawnProdErrCodes.ecInsertFailed, TypeName(Me), "UpdateManfAcceptCommodity"
End If

'run SP for ref table and update the tri state codes
Call RunSP("sp_mfr_accept_part_update", MakeIntSPParm("aLong", alManfID), _
MakeIntSPParm("aLong", alSourceManfID), _
MakeIntSPParm("aLong", -1),
MakeIntSPParm("aLong", alCommodityID), _
MakeIntSPParm("aLong", -1),
MakeStrSPParm("aString", 255, asAcceptInd))

'run SP for pipelines
If asAcceptInd = 1 Then
    Call RunSP("sp_pipeline_mfr_product_add", MakeIntSPParm("aLong", alManfID),

```



```

MakeIntSPParm("aLong", -1), _
MakeIntSPParm("aLong", alCommodityID), _
MakeIntSPParm("aLong", -1))
'run SP to update price
Call UpdatePrice(alManfID, -1, -1, alCommodityID)
'
' Else
' Call RunSP("sp_pipeline_mfr_product_remove", MakeIntSPParm("aLong",
alManfID), _
MakeIntSPParm("aLong", -1), _
MakeIntSPParm("aLong", alCommodityID), _
MakeIntSPParm("aLong", -1))
' End If
'
' Set oDBAccept = Nothing
'
' CtxSetComplete
' Exit Sub

ErrorHandler:
' Set oDBAccept = Nothing
' CtxSetAbort
' RaiseError TypeName(Me), "UpdateManfAcceptCommodity"
' Exit Sub

End Sub
'=====
' this function updates the accept ind for a prod line , its commodities and its
' parts which has been accepted into the
' mfr accept part table
' arguments mfr ID, part ID, source ID, default indicator, available indicator
' returns none
'=====
Public Sub UpdateManfAcceptProdLine(ByVal alManfID As Long, ByVal alSourceManfID As
Long, _
ByVal arrProdLineID As
Variant, ByVal asAcceptInd As String)
On Error GoTo ErrorHandler

Dim sProd As String
Dim lCounter As Long
Dim sSQL As String
Dim sTest As String

For lCounter = 0 To UBound(arrProdLineID)
    If lCounter = UBound(arrProdLineID) Then
        sProd = sProd & CStr(arrProdLineID(lCounter))
    Else
        sProd = sProd & CStr(arrProdLineID(lCounter)) & " , "
    End If
Next
sProd = " ( " & sProd & " ) "

' create temp table for use by stored procedure
sSQL = " CREATE table #parts (part_id int)"
RunSQL (sSQL)

sTest = "before insert"

```

```

        BProd.cls
'      'add all the parts for all the commodities in the array into the temp table
'      sSQL = " INSERT INTO #parts SELECT part_id FROM mfr_accept_part WHERE mfr_id =
" & CStr(alManfID)
'      sSQL = sSQL & " AND src_mfr_id = " & CStr(alSourceManfID) & " AND prod_line_id
in " & sProd
'      RunSQL (sSQL)

      sTest = "before SP"
      'run SP to accept and calc price
      Call RunSP("sp_mfr_accept_part_update_test", MakeIntSPParm("aLong",
alManfID), _
MakeIntSPParm("aLong", alSourceManfID), _
MakeStrSPParm("aString", 255, asAcceptInd), _
MakeStrSPParm("aString", 2, "L"), _
MakeStrSPParm("aString", 1000, sProd))

'      'vars
'      Dim oDBAccept As DawnProd.DBMfrAcceptPart
'      Dim lRow As Long
'
'      Set oDBAccept = CtxCreateObject("DawnProd.DBMfrAcceptPart")
'      lRow = oDBAccept.UpdateMfrAcceptProdLine(alManfID, alSourceManfID,
alProdLineID, asAcceptInd)
'      If lRow <= 0 Then
'          RaiseResError DawnProd.DawnProdErrCodes.ecInsertFailed, TypeName(Me),
"UpdateManfAcceptProdLine"
'      End If
'
'      'run SP for ref table and update the tri state codes
'      Call RunSP("sp_mfr_accept_part_update", MakeIntSPParm("aLong", alManfID), _
MakeIntSPParm("aLong",
alSourceManfID), _
MakeIntSPParm("aLong",
alProdLineID), _
MakeIntSPParm("aLong", -1),
MakeIntSPParm("aLong", -1),
MakeStrSPParm("aString",
255, asAcceptInd))
'
'      'run SP for pipelines
'      If asAcceptInd = 1 Then
'          Call RunSP("sp_pipeline_mfr_product_add", MakeIntSPParm("aLong", alManfID),
MakeIntSPParm("aLong", alProdLineID), _
MakeIntSPParm("aLong", -1), _
MakeIntSPParm("aLong", -1))
'          'run SP to update price
'          Call UpdatePrice(alManfID, -1, alProdLineID, -1)
'
'      Else
'          Call RunSP("sp_pipeline_mfr_product_remove", MakeIntSPParm("aLong",
alManfID), _
MakeIntSPParm("aLong", alProdLineID), _

```

BProd.cls

```

'
MakeIntSPParm("aLong", -1), _
'
MakeIntSPParm("aLong", -1))
'    End If
'    Set oDBAccept = Nothing
'    CtxSetComplete
'    Exit Sub

ErrorHandler:
'    Set oDBAccept = Nothing
'    CtxSetAbort
'    RaiseError TypeName(Me), "UpdateManfAcceptProdLine" & sTest
'    Exit Sub

End Sub
'=====
'this function deletes a part from the mfr accept part table
'arguments - mfr ID, part ID, source mfr ID
'returns none
'=====
Public Sub RemoveManfAcceptPart(ByVal alManfID As Long, ByVal alPartID As Long,
ByVal alSourceManfID As Long)
On Error GoTo ErrorHandler

'vars
Dim oDBAccept As DawnProd.DBMfrAcceptPart
Dim lRow As Long

'run SP for pipelines
Call RunSP("sp_pipeline_mfr_product_remove", MakeIntSPParm("aLong", alManfID), _
MakeIntSPParm("aLong", -1), _
MakeIntSPParm("aLong", -1), _
MakeIntSPParm("aLong", alPartID))

Set oDBAccept = CtxCreateObject("DawnProd.DBMfrAcceptPart")
lRow = oDBAccept.RemoveMfrAcceptPart(alManfID, alPartID, alSourceManfID)
If lRow <= 0 Then
RaiseResError DawnProd.DawnProdErrCodes.ecDeleteFailed, TypeName(Me),
"RemoveManfAcceptPart"
End If

'run SP for ref table and update the tri state codes
Call RunSP("sp_mfr_accept_part_delete", MakeIntSPParm("aLong", alManfID), _
MakeIntSPParm("aLong", -1), _
MakeIntSPParm("aLong", -1), _
MakeIntSPParm("aLong",
alPartID))

CtxSetComplete
Set oDBAccept = Nothing

Exit Sub

ErrorHandler:
CtxSetAbort
Set oDBAccept = Nothing

```

```

                                BProd.cls
RaiseError TypeName(Me), "RemoveManfAcceptPart"
Exit Sub

End Sub
'=====
'this function deletes a commodity and its parts from the mfr accept part table
'arguments mfr ID, commodity ID
'returns none
'=====
Public Sub RemoveManfAcceptCommodity(ByVal aManfID As Long, ByVal aCommodityID As Long)
On Error GoTo ErrHandler

    'vars
    Dim oDBAccept As DawnProd.DBMfrAcceptPart
    Dim lRow As Long

    'run SP for pipelines
    Call RunSP("sp_pipeline_mfr_product_remove", MakeIntSPParm("aLong", aManfID), _
MakeIntSPParm("aLong", -1), _
MakeIntSPParm("aLong", aCommodityID), _
MakeIntSPParm("aLong", -1))

    Set oDBAccept = CtxCreateObject("DawnProd.DBMfrAcceptPart")
    lRow = oDBAccept.RemoveMfrAcceptCommodity(aManfID, aCommodityID)
    If lRow < 0 Then
        RaiseResError DawnProd.DawnProdErrCodes.ecDeleteFailed, TypeName(Me),
"RemoveManfAcceptCommodity"
    End If

    'run SP for ref table and update the tri state codes
    Call RunSP("sp_mfr_accept_part_delete", MakeIntSPParm("aLong", aManfID), _
                                                MakeIntSPParm("aLong", -1),
-
                                                MakeIntSPParm("aLong",
aCommodityID), _
                                                MakeIntSPParm("aLong", -1))

    CtxSetComplete
    Set oDBAccept = Nothing

    Exit Sub

ErrHandler:
    CtxSetAbort
    Set oDBAccept = Nothing
    RaiseError TypeName(Me), "RemoveManfAcceptCommodity"
    Exit Sub

End Sub
'=====
'this function deletes the prod line , its commodity and parts of the mfr accept
part table
'arguments mfr id, prod line id
'returns none
'=====
Public Sub RemoveManfAcceptProdLine(ByVal aManfID As Long, ByVal aProdLineID As Long)
On Error GoTo ErrHandler

```

BProd.cls

```

'vars
Dim oDBAccept As DawnProd.DBMfrAcceptPart
Dim lRow As Long

'run SP for pipelines
Call RunSP("sp_pipeline_mfr_product_remove", MakeIntSPParm("aLong", alManfID), _
MakeIntSPParm("aLong", alProdLineID), _
MakeIntSPParm("aLong", -1), _
MakeIntSPParm("aLong", -1))

Set oDBAccept = CtxCreateObject("DawnProd.DBMfrAcceptPart")
lRow = oDBAccept.RemoveMfrAcceptProdLine(alManfID, alProdLineID)
If lRow < 0 Then
    RaiseResError DawnProd.DawnProdErrCodes.ecDeleteFailed, TypeName(Me),
"RemoveManfAcceptProdLine"
End If

'run SP for ref table and update the tri state codes
Call RunSP("sp_mfr_accept_part_delete", MakeIntSPParm("aLong", alManfID), _
MakeIntSPParm("aLong", _
alProdLineID), _
MakeIntSPParm("aLong", -1),
-
MakeIntSPParm("aLong", -1))

CtxSetComplete
Set oDBAccept = Nothing

Exit Sub

ErrorHandler:
CtxSetAbort
Set oDBAccept = Nothing
RaiseError TypeName(Me), "RemoveManfAcceptProdLine"
Exit Sub

End Sub
'=====
'this function adds a part to the org avail part table - mfr allows aprt to be seen
by this org
'arguments - org ID, part ID, commodity ID, prod line ID
'returns none
'=====
Public Sub AddOrgAvailPart(ByVal alOrgID As Long, ByVal alPartID As Long, ByVal
alCommodityID As Long, _
ByVal alProdLineID As Long)
On Error GoTo ErrorHandler

'vars
Dim oDBOrg As DawnProd.DBOrgAvailPart
Dim lRow As Long

'remove and add part
Set oDBOrg = CtxCreateObject("DawnProd.DBOrgAvailPart")
lRow = oDBOrg.RemoveOrgAvailPart(alOrgID, alPartID)
If lRow < 0 Then
    RaiseResError DawnProd.DawnProdErrCodes.ecDeleteFailed, TypeName(Me),
"AddOrgAvailPart"

```

BProd.cls

```

End If
lRow = oDBOrg.AddOrgAvailPart(alOrgID, alPartID, alCommodityID, alProdLineID)
If lRow <= 0 Then
    RaiseResError DawnProd.DawnProdErrCodes.ecInsertFailed, TypeName(Me),
"AddOrgAvailPart"
End If

'run SP for ref table and update the tri state codes
Call RunSP("sp_org_avail_part_insert", MakeIntSPParm("aLong", alOrgID), _
                                                MakeIntSPParm("aLong", -1),
-                                                MakeIntSPParm("aLong", -1),
-                                                MakeIntSPParm("aLong",
alPartID))

'run SP for pipelines
Call RunSP("sp_pipeline_org_product_add", MakeIntSPParm("aLong", alOrgID), _
MakeIntSPParm("aLong", -1), _
MakeIntSPParm("aLong", -1), _
MakeIntSPParm("aLong", alPartID))

CtxSetComplete
Set oDBOrg = Nothing

Exit Sub

ErrorHandler:
CtxSetAbort
Set oDBOrg = Nothing
RaiseError TypeName(Me), "AddOrgAvailPart"
Exit Sub
End Sub
'=====
'this function adds a commodity and its parts to the org avial part table - mfr
allows commodity to be seen by this org
'arguments - org ID, commodity ID
'returns none
'=====
Public Sub AddOrgAvailCommodity(ByVal alOrgID As Long, ByVal alCommodityID As Long)
On Error GoTo ErrorHandler

'vars
Dim oDBOrg As DawnProd.DBOrgAvailPart
Dim lRow As Long

'remove all org avail parts for this commodity first and then all parts for this
commodity

Set oDBOrg = CtxCreateObject("DawnProd.DBOrgAvailPart")
lRow = oDBOrg.RemoveOrgAvailCommodity(alOrgID, alCommodityID)
If lRow < 0 Then
    RaiseResError DawnProd.DawnProdErrCodes.ecDeleteFailed, TypeName(Me),
"AddOrgAvailCommodity"
End If
lRow = oDBOrg.AddOrgAvailCommodity(alOrgID, alCommodityID)
If lRow <= 0 Then
    RaiseResError DawnProd.DawnProdErrCodes.ecInsertFailed, TypeName(Me),
"AddOrgAvailCommodity"
End If

```

BProd.cls

```

'run SP for ref table and update the tri state codes
Call RunSP("sp_org_avail_part_insert", MakeIntSPParm("aLong", aOrgID), _
                                                MakeIntSPParm("aLong", -1),
-
                                                MakeIntSPParm("aLong",
aCommodityID), _
                                                MakeIntSPParm("aLong", -1))

'run SP for pipelines
Call RunSP("sp_pipeline_org_product_add", MakeIntSPParm("aLong", aOrgID), _
MakeIntSPParm("aLong", -1), _
MakeIntSPParm("aLong", aCommodityID), _
MakeIntSPParm("aLong", -1))

CtxSetComplete
Set oDBOrg = Nothing

Exit Sub

ErrorHandler:
CtxSetAbort
Set oDBOrg = Nothing
RaiseError TypeName(Me), "AddOrgAvailCommodity"
Exit Sub
End Sub
'=====
'this function adds a prod line, its commodities and its parts to the org avail
part table - mfr allows prod line to be seen by this org
'arguments - org ID, prod line ID
'returns none
'=====
Public Sub AddOrgAvailProdLine(ByVal aOrgID As Long, ByVal aProdLineID As Long)
On Error GoTo ErrorHandler

'vars
Dim oDBOrg As DawnProd.DBOrgAvailPart
Dim lRow As Long

'remove all existing parts for the prodline first and then add all parts for
this prod line
Set oDBOrg = CtxCreateObject("DawnProd.DBOrgAvailPart")
lRow = oDBOrg.RemoveOrgAvailProdLine(aOrgID, aProdLineID)
If lRow < 0 Then
RaiseResError DawnProd.DawnProdErrCodes.ecDeleteFailed, TypeName(Me),
"AddOrgAvailProdLine"
End If
lRow = oDBOrg.AddOrgAvailProdLine(aOrgID, aProdLineID)
If lRow <= 0 Then
RaiseResError DawnProd.DawnProdErrCodes.ecInsertFailed, TypeName(Me),
"AddOrgAvailProdLine"
End If

'run SP for ref table and update the tri state codes
Call RunSP("sp_org_avail_part_insert", MakeIntSPParm("aLong", aOrgID), _
                                                MakeIntSPParm("aLong",
aProdLineID), _
                                                MakeIntSPParm("aLong", -1),
-
                                                MakeIntSPParm("aLong", -1))

```

```

                                BProd.cls

'run SP for pipelines
Call RunSP("sp_pipeline_org_product_add", MakeIntSPParm("aLong", aOrgID), _
MakeIntSPParm("aLong", aProdLineID), _
MakeIntSPParm("aLong", -1), _
MakeIntSPParm("aLong", -1))

    CtxSetComplete
    Set oDBOrg = Nothing

Exit Sub

ErrorHandler:
    CtxSetAbort
    Set oDBOrg = Nothing
    RaiseError TypeName(Me), "AddOrgAvailProdLine"
    Exit Sub
End Sub
'=====
'this function removes a part from the org avail part table - mfr removes it from
org listing
'arguments - org ID, part ID
'returns none
'=====
Public Sub RemoveOrgAvailPart(ByVal aOrgID As Long, ByVal aPartID As Long)
On Error GoTo ErrorHandler

    'vars
    Dim oDBOrg As DawnProd.DBOrgAvailPart
    Dim lRow As Long

    'run SP for pipelines
    Call RunSP("sp_pipeline_org_product_remove", MakeIntSPParm("aLong", aOrgID), _
MakeIntSPParm("aLong", -1), _
MakeIntSPParm("aLong", -1), _
MakeIntSPParm("aLong", aPartID))

    Set oDBOrg = CtxCreateObject("DawnProd.DBOrgAvailPart")
    lRow = oDBOrg.RemoveOrgAvailPart(aOrgID, aPartID)
    If lRow < 0 Then
        RaiseResError DawnProd.DawnProdErrCodes.ecDeleteFailed, TypeName(Me),
"RemoveOrgAvailPart"
    End If

    'run SP for ref table and update the tri state codes
    Call RunSP("sp_org_avail_part_delete", MakeIntSPParm("aLong", aOrgID), _
MakeIntSPParm("aLong", -1), _
-
MakeIntSPParm("aLong", -1), _
-
MakeIntSPParm("aLong",
aPartID))

    CtxSetComplete
    Set oDBOrg = Nothing

```



```

Exit Sub

ErrHandler:
  CtxSetAbort
  Set oDBOrg = Nothing
  RaiseError TypeName(Me), "RemoveOrgAvailPart"
  Exit Sub
End Sub
'=====
'this function removes a commodity from the org avail part table - mfr removes it
from org listing
'arguments - org ID, commodity ID
'returns none
'=====
Public Sub RemoveOrgAvailCommodity(ByVal aOrgID As Long, ByVal aCommodityID As
Long)
On Error GoTo ErrHandler

  'vars
  Dim oDBOrg As DawnProd.DBOrgAvailPart
  Dim lRow As Long

  'run SP for pipelines
  Call RunSP("sp_pipeline_org_product_remove", MakeIntSPParm("aLong", aOrgID), _
MakeIntSPParm("aLong", -1), _
MakeIntSPParm("aLong", aCommodityID), _
MakeIntSPParm("aLong", -1))

  Set oDBOrg = CtxCreateObject("DawnProd.DBOrgAvailPart")
  lRow = oDBOrg.RemoveOrgAvailCommodity(aOrgID, aCommodityID)
  If lRow < 0 Then
    RaiseResError DawnProd.DawnProdErrCodes.ecDeleteFailed, TypeName(Me),
"RemoveOrgAvailCommodity"
  End If

  'run SP for ref table and update the tri state codes
  Call RunSP("sp_org_avail_part_delete", MakeIntSPParm("aLong", aOrgID), _
MakeIntSPParm("aLong", -1),
_
MakeIntSPParm("aLong",
aCommodityID), _
MakeIntSPParm("aLong", -1))

  CtxSetComplete
  Set oDBOrg = Nothing

  Exit Sub

ErrHandler:
  CtxSetAbort
  Set oDBOrg = Nothing
  RaiseError TypeName(Me), "RemoveOrgAvailCommodity"
  Exit Sub
End Sub
'=====
'this function removes a prodline, its commodities and parts from the org avail part
table - mfr removes it from org listing
'arguments - org ID, prod line ID

```

'returns none

```

=====
Public Sub RemoveOrgAvailProdLine(ByVal aOrgID As Long, ByVal aProdLineID As Long)
On Error GoTo ErrHandler

```

'vars

```

Dim oDBOrg As DawnProd.DBOrgAvailPart
Dim lRow As Long

```

'run SP for pipelines

```

Call RunSP("sp_pipeline_org_product_remove", MakeIntSPParm("aLong", aOrgID), _

```

```

MakeIntSPParm("aLong", aProdLineID), _

```

```

MakeIntSPParm("aLong", -1), _

```

```

MakeIntSPParm("aLong", -1))

```

```

Set oDBOrg = CtxCreateObject("DawnProd.DBOrgAvailPart")

```

```

lRow = oDBOrg.RemoveOrgAvailProdLine(aOrgID, aProdLineID)

```

```

If lRow < 0 Then

```

```

    RaiseResError DawnProd.DawnProdErrCodes.ecDeleteFailed, TypeName(Me),

```

```

    "RemoveOrgAvailProdLine"

```

```

End If

```

'run SP for ref table and update the tri state codes

```

Call RunSP("sp_org_avail_part_delete", MakeIntSPParm("aLong", aOrgID), _
MakeIntSPParm("aLong",

```

```

aProdLineID), _

```

```

MakeIntSPParm("aLong", -1),

```

```

MakeIntSPParm("aLong", -1))

```

```

CtxSetComplete

```

```

Set oDBOrg = Nothing

```

```

Exit Sub

```

ErrHandler:

```

CtxSetAbort

```

```

Set oDBOrg = Nothing

```

```

RaiseError TypeName(Me), "RemoveOrgAvailProdLine"

```

```

Exit Sub

```

End Sub

```

=====
'this function adds a part to the default avial part table - mfr allows aprt to be
seen by non org members

```

```

'arguments - mfr ID, part ID, commodity ID, prod line ID

```

```

'returns none

```

```

=====
Public Sub AddDfltAvailPart(ByVal aManfID As Long, ByVal aPartID As Long, ByVal
aCommodityID As Long, _

```

```

ByVal aProdLineID As Long)

```

```

On Error GoTo ErrHandler

```

'vars

```

Dim oDBDflt As DawnProd.DBDfltAvailPart

```

```

Dim lRow As Long

```

'remove the part if it already exists and re add the part

```

Set oDBDflt = CtxCreateObject("DawnProd.DBDfltAvailPart")

```

```

lRow = oDBDflt.RemoveDfltAvailPart(aManfID, aPartID)

```

BProd.cls

```

If lRow < 0 Then
    RaiseResError DawnProd.DawnProdErrCodes.ecDeleteFailed, TypeName(Me),
"AddDfltAvailPart"
End If
lRow = oDBDflt.AddDfltAvailPart(alManfID, alPartID, alCommodityID, alProdLineID)
If lRow <= 0 Then
    RaiseResError DawnProd.DawnProdErrCodes.ecInsertFailed, TypeName(Me),
"AddDfltAvailPart"
End If

'run SP for ref table and update the tri state codes
Call RunSP("sp_dflt_avail_part_insert", MakeIntSPParm("aLong", alManfID), _
                                                MakeIntSPParm("aLong", -1),
-
                                                MakeIntSPParm("aLong", -1),
-
                                                MakeIntSPParm("aLong",
alPartID))

CtxSetComplete
Set oDBDflt = Nothing

Exit Sub

ErrorHandler:
CtxSetAbort
Set oDBDflt = Nothing
RaiseError TypeName(Me), "AddDfltAvailPart"
Exit Sub
End Sub
'=====
'this function adds a commodity and its parts to the default avial part table - mfr
allows commodity to be seen by non org members
'arguments - mfr ID, commodity ID
'returns none
'=====
Public Sub AddDfltAvailCommodity(ByVal alManfID As Long, ByVal alCommodityID As
Long)
On Error GoTo ErrorHandler

'vars
Dim oDBDflt As DawnProd.DBDFltAvailPart
Dim lRow As Long

'remove existing parts for this commodity and then add all parts in this
commodity
Set oDBDflt = CtxCreateObject("DawnProd.DBDFltAvailPart")
lRow = oDBDflt.RemoveDfltAvailCommodity(alManfID, alCommodityID)
If lRow < 0 Then
    RaiseResError DawnProd.DawnProdErrCodes.ecDeleteFailed, TypeName(Me),
"AddDfltAvailCommodity"
End If
lRow = oDBDflt.AddDfltAvailCommodity(alManfID, alCommodityID)
If lRow <= 0 Then
    RaiseResError DawnProd.DawnProdErrCodes.ecInsertFailed, TypeName(Me),
"AddDfltAvailCommodity"
End If

'run SP for ref table and update the tri state codes
Call RunSP("sp_dflt_avail_part_insert", MakeIntSPParm("aLong", alManfID), _
                                                MakeIntSPParm("aLong", -1),
-
                                                MakeIntSPParm("aLong",

```

```

alCommodityID), _
                                                    MakeIntSPParm("aLong", -1))

    CtxSetComplete
    Set oDBDflt = Nothing

    Exit Sub

ErrorHandler:
    CtxSetAbort
    Set oDBDflt = Nothing
    RaiseError TypeName(Me), "AddDfltAvailCommodity"
    Exit Sub
End Sub
'=====
'this function adds a prod line, its commodities and its parts to the default avial
part table
'mfr allows prod line to be seen by non org members
'arguments - mfr ID, prod line ID
'returns none
'=====
Public Sub AddDfltAvailProdLine(ByVal alManfID As Long, ByVal alProdLineID As Long)
On Error GoTo ErrorHandler

    'vars
    Dim oDBDflt As DawnProd.DBDFltAvailPart
    Dim lRow As Long

    'remove all existing parts for this prod line and then add the whole prod line
    Set oDBDflt = CtxCreateObject("DawnProd.DBDFltAvailPart")
    lRow = oDBDflt.RemovedfltAvailProdLine(alManfID, alProdLineID)
    If lRow < 0 Then
        RaiseResError DawnProd.DawnProdErrCodes.ecDeleteFailed, TypeName(Me),
"AddDfltAvailProdLine"
    End If
    lRow = oDBDflt.AddDfltAvailProdLine(alManfID, alProdLineID)
    If lRow <= 0 Then
        RaiseResError DawnProd.DawnProdErrCodes.ecInsertFailed, TypeName(Me),
"AddDfltAvailProdLine"
    End If

    'run SP for ref table and update the tri state codes
    Call RunSP("sp_dflt_avail_part_insert", MakeIntSPParm("aLong", alManfID), _
                                                    MakeIntSPParm("aLong",
alProdLineID), _
                                                    MakeIntSPParm("aLong", -1),
-
                                                    MakeIntSPParm("aLong", -1))

    CtxSetComplete
    Set oDBDflt = Nothing

    Exit Sub

ErrorHandler:
    CtxSetAbort
    Set oDBDflt = Nothing
    RaiseError TypeName(Me), "AddDfltAvailProdLine"
    Exit Sub
End Sub
'=====
'this function removes a part from the default avail part table - mfr removes it
from non org member display
'arguments - mfr ID, part ID

```

```

'returns none
'=====
Public Sub RemovedfltAvailPart(ByVal aManfID As Long, ByVal aPartID As Long)
On Error GoTo ErrHandler

'vars
Dim oDBDflt As DawnProd.DBDFltAvailPart
Dim lRow As Long

Set oDBDflt = CtxCreateObject("DawnProd.DBDFltAvailPart")
lRow = oDBDflt.RemovedfltAvailPart(aManfID, aPartID)
If lRow <= 0 Then
    RaiseResError DawnProd.DawnProdErrCodes.ecDeleteFailed, TypeName(Me),
"RemovedfltAvailPart"
End If

'run SP for ref table and update the tri state codes
Call RunSP("sp_dflt_avail_part_delete", MakeIntSPParm("aLong", aManfID), _
                                                MakeIntSPParm("aLong", -1),
-
                                                MakeIntSPParm("aLong", -1),
-
                                                MakeIntSPParm("aLong",
aPartID))

CtxSetComplete
Set oDBDflt = Nothing

Exit Sub

ErrHandler:
CtxSetAbort
Set oDBDflt = Nothing
RaiseError TypeName(Me), "RemovedfltAvailPart"
Exit Sub
End Sub
'=====
'this function removes a commodity from the default avail part table - mfr removes
it from member listing
'arguments - mfr ID, commodity ID
'returns none
'=====
Public Sub RemovedfltAvailCommodity(ByVal aManfID As Long, ByVal aCommodityID As
Long)
On Error GoTo ErrHandler

'vars
Dim oDBDflt As DawnProd.DBDFltAvailPart
Dim lRow As Long

Set oDBDflt = CtxCreateObject("DawnProd.DBDFltAvailPart")
lRow = oDBDflt.RemovedfltAvailCommodity(aManfID, aCommodityID)
If lRow < 0 Then
    RaiseResError DawnProd.DawnProdErrCodes.ecDeleteFailed, TypeName(Me),
"RemovedfltAvailCommodity"
End If

'run SP for ref table and update the tri state codes
Call RunSP("sp_dflt_avail_part_delete", MakeIntSPParm("aLong", aManfID), _
                                                MakeIntSPParm("aLong", -1),
-
                                                MakeIntSPParm("aLong",
aCommodityID), _

```

```

    CtxSetComplete
    Set oDBDflt = Nothing

    Exit Sub

ErrorHandler:
    CtxSetAbort
    Set oDBDflt = Nothing
    RaiseError TypeName(Me), "RemovedfltAvailCommodity"
    Exit Sub
End Sub
'=====
'this function removes a prodline, its commodities and parts from the default avail
part table - mfr removes it from member listing
'arguments - mfr ID, prod line ID
'returns none
'=====
Public Sub RemovedfltAvailProdLine(ByVal alManfID As Long, ByVal alProdLineID As
Long)
On Error GoTo ErrorHandler

    'vars
    Dim oDBDflt As DawnProd.DBDFltAvailPart
    Dim lRow As Long

    Set oDBDflt = CtxCreateObject("DawnProd.DBDFltAvailPart")
    lRow = oDBDflt.RemovedfltAvailProdLine(alManfID, alProdLineID)
    If lRow < 0 Then
        RaiseResError DawnProd.DawnProdErrCodes.ecDeleteFailed, TypeName(Me),
"RemovedfltAvailProdLine"
    End If

    'run SP for ref table and update the tri state codes
    Call RunSP("sp_dflt_avail_part_delete", MakeIntSPParm("aLong", alManfID), _
                                                    MakeIntSPParm("aLong",
alProdLineID), _
                                                    MakeIntSPParm("aLong", -1),
-
                                                    MakeIntSPParm("aLong", -1))

    CtxSetComplete
    Set oDBDflt = Nothing

    Exit Sub

ErrorHandler:
    CtxSetAbort
    Set oDBDflt = Nothing
    RaiseError TypeName(Me), "RemovedfltAvailProdLine"
    Exit Sub
End Sub
'=====
'this function nulls the color from the part
'args part ID, last maint dtm
'returns long
'=====
Public Sub RemPartColor(ByVal alPartID, ByVal adMaintDate As Date)
On Error GoTo ErrorHandler

    Dim oDB As DawnProd.DBPart

```

BProd.cls

```

Dim lRow As Long

Set oDB = CtxCreateObject("DawnProd.DBPart")
lRow = oDB.RemPartColor(alPartID, adMaintDate)
If lRow <> 1 Then
    RaiseResError DawnProd.DawnProdErrCodes.ecDeleteFailed, TypeName(Me),
"RemPartColor"
End If
CtxSetComplete
Set oDB = Nothing
Exit Sub

ErrorHandler:
CtxSetAbort
Set oDB = Nothing
RaiseError TypeName(Me), "RemPartColor"
Exit Sub

End Sub
'=====
'this function nulls the stamp face for a commodity
'args - commodity ID, last maint date
'returns long - no of rows affected
'=====
Public Sub RemCommStampface(ByVal alCommodityID As Long, ByVal adMaintDate As Date)
On Error GoTo ErrorHandler

'vars
Dim oDB As DawnProd.DBComm
Dim lRow As Long

Set oDB = CtxCreateObject("DawnProd.DBComm")
lRow = oDB.RemCommStampface(alCommodityID, adMaintDate)

If lRow <> 1 Then
    RaiseResError DawnProd.DawnProdErrCodes.ecDeleteFailed, TypeName(Me),
"RemCommStampface"
End If
CtxSetComplete
Set oDB = Nothing
Exit Sub

ErrorHandler:
CtxSetAbort
Set oDB = Nothing
RaiseError TypeName(Me), "RemCommStampface"
Exit Sub

End Sub
'=====
'this function calls the price update procedure with relavant parameters
'returns none
'=====
Public Sub UpdatePrice(ByVal alManfID As Long, ByVal alOrgID As Long, _
ByVal alProdLineID As Long, ByVal alCommodityID As Long)
On Error GoTo ErrorHandler

Dim sSQL As String
Dim oRS As ADODB.Recordset

```

BProd.cls

'call the procedure for updating ref table prices

```
Call RunSP("sp_update_price", MakeIntSPParm("aLong", alManfID), _  
MakeIntSPParm("aLong", alOrgID), _  
MakeIntSPParm("aLong", alProdLineID), _  
MakeIntSPParm("aLong", alCommodityID))
```

'call procedure for all orgs which have this prod line also, if mfr is being updated

```
If alManfID > 0 Then  
    sSQL = " SELECT DISTINCT org.org_id FROM org , org_prod_ref WHERE  
org.mfr_id = " & CStr(alManfID)  
    sSQL = sSQL & " AND org.org_id = org_prod_ref.org_id "  
    If alProdLineID > 0 Then  
        sSQL = sSQL & " AND org_prod_ref.prod_line_id = " &  
CStr(alProdLineID)  
    ElseIf alCommodityID > 0 Then  
        sSQL = sSQL & " AND org_prod_ref.commodity_id = " &  
CStr(alCommodityID)  
    End If  
    Set ORS = GetSQLRecordset(sSQL)  
  
    If Not ORS.EOF Then  
        ORS.MoveFirst  
        Do While Not ORS.EOF  
            Call RunSP("sp_update_price",  
MakeIntSPParm("aLong", -1), _  
MakeIntSPParm("aLong", CLng(ORS("org_id"))),  
_  
MakeIntSPParm("aLong", alProdLineID), _  
MakeIntSPParm("aLong", alCommodityID))  
            ORS.MoveNext  
        Loop  
    End If  
End If  
  
Set ORS = Nothing  
CtxSetComplete  
Exit Sub
```

ErrHandler:

```
Set ORS = Nothing  
CtxSetAbort  
RaiseError TypeName(Me), "UpdatePrice"  
Exit Sub
```

End Sub

```
'=====
```

'deletes pricing information for a part associated with a manufacturer.
'returns none

```
'=====
```

Public Sub DelManfPartPr(ByVal alManfID As Long, ByVal alPartID As Long, ByVal adLastMaintdate As Date)

On Error GoTo ErrHandler

'vars

```
Dim oDB As DawnProd.DBPartPricing  
Dim lRow As Long
```

'call DB function


```

                                BProd.cls
Set oDB = CtxCreateObject("DawnProd.DBPartPricing")
lRow = oDB.DelManfPartPr(alManfID, alPartID, adLastMaintdate)
If lRow < 0 Then
    RaiseResError DawnProd.DawnProdErrCodes.ecDeleteFailed, TypeName(Me),
"DelManfPartPr"
End If

CtxSetComplete
Set oDB = Nothing
Exit Sub

ErrorHandler:
CtxSetAbort
Set oDB = Nothing
RaiseError TypeName(Me), "DelManfPartPr"
Exit Sub

End Sub

'=====
'
'deletes pricing information for a commodity associated with a manufacturer.
'returns none
'=====
Public Sub DelManfCommPr(ByVal alManfID As Long, ByVal alCommodityID As Long, ByVal
adLastMaintdate As Date)
    On Error GoTo ErrorHandler

    'vars
    Dim oDB As DawnProd.DBCommPricing
    Dim lRow As Long

    'call DB function
    Set oDB = CtxCreateObject("DawnProd.DBCommPricing")
    lRow = oDB.DelManfCommPr(alManfID, alCommodityID, adLastMaintdate)
    If lRow < 0 Then
        RaiseResError DawnProd.DawnProdErrCodes.ecDeleteFailed, TypeName(Me),
"DelManfCommPr"
    End If

    CtxSetComplete
    Set oDB = Nothing
    Exit Sub

ErrorHandler:
CtxSetAbort
Set oDB = Nothing
RaiseError TypeName(Me), "DelManfCommPr"
Exit Sub

End Sub

'=====
'
'deletes pricing information for a prod line associated with a manufacturer.
'returns none
'=====
Public Sub DelManfProdLinePr(ByVal alManfID As Long, ByVal alProdLineID As Long,
ByVal adLastMaintdate As Date)
    On Error GoTo ErrorHandler

```

BProd.cls

```

'vars
Dim oDB As DawnProd.DBProdLinePricing
Dim lRow As Long

'call DB function
Set oDB = CtxCreateObject("DawnProd.DBProdLinePricing")
lRow = oDB.DelManfProdLinePr(alManfID, alProdLineID, adLastMaintdate)
If lRow < 0 Then
    RaiseResError DawnProd.DawnProdErrCodes.ecDeleteFailed, TypeName(Me),
"DelManfProdLinePr"
End If

CtxSetComplete
Set oDB = Nothing
Exit Sub

ErrorHandler:
CtxSetAbort
Set oDB = Nothing
RaiseError TypeName(Me), "DelManfProdLinePr"
Exit Sub

End Sub

'=====
'this function subscribes a mfr to a pipeline and then adds all the pipeline parts
to the accept table
'mfr ID, subscription code
'returns none
'=====
Public Sub PipelineSubscribe(ByVal alManfID As Long, ByVal asSubCode As String,
ByVal alSecurityCode As Long)
    On Error GoTo ErrorHandler

    'vars
    Dim oOrg As Object
    Dim lOrg As Long
    Dim oCtx As ObjectContext
    Dim oRS As ADODB.Recordset

    'disable commit
    Set oCtx = GetObjectContext()
    If Not oCtx Is Nothing Then
        oCtx.DisableCommit
    End If

    'create org and set subscribe
    Set oOrg = CtxCreateObject("DawnOrg.BOrg")
    Set oRS = oOrg.PipelineSubscribe(alManfID, asSubCode, alSecurityCode)

    If oRS.EOF Then
        RaiseResError DawnProd.DawnProdErrCodes.ecSubscribeFailed, TypeName(Me),
"PipelineSubscribe"
    End If

    'unsubscribe if the mfr was a single source mfr
    If Not IsNull(oRS("old_org_id")) Then
        lOrg = oRS("old_org_id")
        If lOrg > 0 Then
            Call RunSP("sp_pipeline_org_remove", MakeIntSPParm("aLong", lOrg))
        End If
    End If
End Sub

```

BProd.cls

```

'run the SP for subscribe
lOrg = ORS("org_id")
Call RunSP("sp_pipeline_org_add", MakeIntSPParm("aLong", lOrg))

'end function and enable commit
If Not oCtx Is Nothing Then
    oCtx.EnableCommit
End If
CtxSetComplete
Set oOrg = Nothing
Set oCtx = Nothing
Set oRS = Nothing
Exit Sub

ErrHandler:
'enable commit
If Not oCtx Is Nothing Then
    oCtx.EnableCommit
End If
CtxSetAbort
Set oOrg = Nothing
Set oCtx = Nothing
Set oRS = Nothing
RaiseError TypeName(Me), "PipelineSubscribe"
End Sub

'=====
'this function subscribes a mfr to a pipeline and then adds all the pipeline parts
to the accept table
'mfr ID, subscription code
'returns none
'this function is not used anymore - uday 3/6/00
'=====
Public Sub PipelineUnsubscribe(ByVal alManfID As Long, ByVal alSrcManfID As Long)
    On Error GoTo ErrHandler
    'vars
    Dim oOrg As Object
    Dim oRS As ADODB.Recordset
    Dim sSQL As String
    Dim lOrg As Long
    Dim oCtx As ObjectContext
    'disable commit
    Set oCtx = GetObjectContext()
    If Not oCtx Is Nothing Then
        oCtx.DisableCommit
    End If
    'create org and set subscribe
    Set oOrg = CtxCreateObject("DawnOrg.BOrg")
    If Not oOrg.PipelineUnsubscribe(alManfID) Then
        RaiseResError DawnProd.DawnProdErrCodes.ecUnSubscribeFailed, TypeName(Me),
        "PipelineUnsubscribe"
    End If
    'get pipeline org
    sSQL = "SELECT mfr_id , org_id , pl_mfr_id FROM org "
    sSQL = sSQL & " WHERE mfr_id = " & CStr(alSrcManfID)
    sSQL = sSQL & " AND pl_mfr_id = " & CStr(alManfID)
    Set oRS = GetSQLRecordset(sSQL)

```

BProd.cls

```

'run the SP for subscribe
If oRs.RecordCount > 0 Then
    lOrg = oRs("org_id")
    Call RunSP("sp_pipeline_org_remove", MakeIntSPParm("aLong", lOrg))
End If

'end function and enable commit
If Not oCtx Is Nothing Then
    oCtx.EnableCommit
End If
CtxSetComplete
Set oOrg = Nothing
Set oCtx = Nothing
Exit Sub

'ErrorHandler:
If Not oCtx Is Nothing Then
    oCtx.EnableCommit
End If
CtxSetAbort
Set oOrg = Nothing
Set oCtx = Nothing
RaiseError TypeName(Me), "PipelineUnsubscribe"
'End Sub

'=====
'this function sets the default mfr source for a product using a procedure
'args - mfr ID, source mfr id, prodline ID or commodity ID or part ID
'returns none if no errpr
'=====
Public Sub UpdateDefaultSource(ByVal alManfID As Long, ByVal alSourceManfID As Long,
ByVal alProdLineID As Long, _
ByVal alCommodityID As Long, ByVal
alPartID As Long)
    On Error GoTo ErrHandler

    'set up IDs
    If alProdLineID > 0 Then
        alCommodityID = -1
        alPartID = -1
    End If
    If alCommodityID > 0 Then
        alProdLineID = -1
        alPartID = -1
    End If
    If alPartID > 0 Then
        alProdLineID = -1
        alCommodityID = -1
    End If

    'run SP
    Call RunSP("sp_set_default_source", MakeIntSPParm("aLong", alManfID), _
alSourceManfID), _
MakeIntSPParm("aLong",
alProdLineID), _
MakeIntSPParm("aLong",
alCommodityID), _
MakeIntSPParm("aLong",
alPartID))

    'exit
    CtxSetComplete

```

BProd.cls

```
Exit Sub

ErrorHandler:
    CtxSetAbort
    RaiseError TypeName(Me), "UpdateDefaultSource"
    Exit Sub
End Sub
```